# Computer Science Department

# TECHNICAL REPORT



**Learning as the Evolution of Representation**

*Pasquale Caianiello*

**Technical Report 477**

November 1989

## NEW YORK UNIVERSITY

Department of Computer Science
Courant Institute of Mathematical Sciences
251 MERCER STREET, NEW YORK, N.Y. 10012

# Learning as the Evolution of Representation

*Pasquale Caianiello*

**Technical Report 477**

November 1989

# Learning as the Evolution of Representation*

Pasquale Caianiello

**Abstract**

A widely accepted approach to the formal modelling of learning is to view it as a process of constructing high-level structures that allow compression of raw data. The main results of this thesis are a number of precise formulation of this process, and a few algorithms that carry it out.

Our rigorous definitions allow us to identify of a novel concept of structure: the *alphabet*. We propose that a learning mechanism is one which constructs such structures as representations of the given data. The alphabet allows to draw interesting links between theories in pattern recognition and pattern classification.

The structural descriptions constructed by the algorithms are based on the two concepts of *code* and *classification* used, as data compression tools. We provide means for evaluating the efficiency of the encodings created using ideas from information theory. We reduce learning to an optimization problem and we suggest that the mechanisms proposed work at any descriptive level.

Our approach was inspired by the goal of interpreting learning as the natural evolution of a physical system to its ground states.

The applications discussed include an architecture and a learning rule for neural nets as well as a procedure for grammatical inference. Some of the ideas proposed have been tested on meaningful examples.

*This research has been supported in part by CNR bando n. 203.1.34 and NSF grant #IRI-8801529

## Acknowledgement

Writing the acknowledgement to a Ph.D. thesis is like being on TV for the first time: it's probably all most people are going to read and you cannot help waving at everybody. However, I'll keep my list limited to the ones who influenced my work directly, I thank all the other friends collectively.

Inevitably, this thesis developed through my dialogues with many people. It grew up with their comments, supports, objections. First of all I wish to thank my advisor Professor Ernest Davis. We carried on together the hard part of the work, the one in which ideas have to face the expressive limits of available formal resources. The best measure of how good our dialogue has been is in the valuation of its result. I am very grateful for the help, good advise, and support he gave me, which went well beyond the mere technical sphere.

I received a complementary help from my second advisor Professor Bhubaneswar Mishra. His recommendation always pointed me in the right direction. Once again, I was asking for something more than just technicalities; I wish to thank him for his support.

I would like to thank Professor Eduardo Caianiello for the embryonic ideas he gave me and for making me understand that sometimes it is easier to find the solution to a problem by solving a more general case. I'm also indebted to Professor Marco Fontana from whom I got the first imprinting on how to write a thesis.

Isabeau Birindelli suggested numerous improvements, fixed several mathematical steps, and helped me in many other ways. Talking to her inspired many good ideas. Professor Dennis Shasha followed the development of this thesis and helped to make it stronger. Alex Botta gave good comments on previous drafts. Special thanks to Professor Ralph Grishman for making both his time and resources frequently available. Professors Marin Davis, Naomi Sager, and Tomasz Strzalkowski had been very kind to answer my questions.

Anne Dinning helped me with her friendship and her skill, whenever I needed.

I had many clarifying conversations with Filippo Cesi, Vincenzo Cutello, Piero De Chiara, Giovanni Gallo, Cristiano Husu, Marco Isopi, Leo Joskowicz, Piera Morani, Stefano Olla, Alberto Policriti, and Ofer Zajicek. They helped me with their expertise.

Courant Institute made my long stay in New York both easy and difficult in a way which got my work done. Roberto Tagliaferri and other friends at the Universitá di Salerno and IIASS gave me important feedback and technical help. The stimulating interdisciplinary environment of the 1988 Summer School of Complex Systems of the Santa Fé Institute favored the development of many ideas.

This final paragraph is devoted to a few close friends some already mentioned before. Their names have been omitted because listing them would impose a total order that does not exist. The place of honor is for my family, my brothers and my parents, to whom I dedicate this work.

# Contents

"It is an old idea that the more pointedly and logically we formulate a thesis, the more irresistibly it cries out for its antithesis"

*Hermann Hesse*
The Glass Bead Game

# Chapter 1

# The Problem

## 1.1 Introduction

The story starts one day in Olympus when Zeus has a headache. He calls up Hephaestus, the hard working locksmith god, who splits his head in half and out comes Athena, the goddess of Reason and War. That day Hephaestus invents divide&conquer, reason, and war all at once. Since then mankind have been using them to put Zeus' head back together and reconstruct the big picture.

This thesis is about learning and inductive inference. It is the result of three years of work on the subject with the goal of arriving at a general understanding of the concepts in its attempt to reconstruct the big picture for the fragmented reality of the community interested in those subjects. To do this, it had to be "general", i.e., "applicable to the widest range of contexts and situations". Nonetheless it aimed at a formal comprehension: the task was to be considered complete when the understanding reached a mathematical definition of the terms involved, and when the guidelines for a computer implementations seemed sufficiently solid.

In evaluating it, the reader should consider that learning theory yields problems and results which shed light on many fundamental topics of today's scientific quest. Inductive inference is a paradigm to model several processes like the search for physical laws, Popper's logic of discovery [6], the adaptation of a biological being to its environment [66], intelligence, evolution. We hope the implications that its

results may have will contribute to keep the interest in learning theory alive.

### Pragmatic Applications

During my carreer as a student I have been dealing with computers in many ways. Though I have spent great efforts in adapting to their and their appendages' needs, I feel that they have made no effort to do the same with me. After using the same few operating system commands and addressing the same few files over and over again I still get a "not found" answer when I mispell it by a single unmeaningful letter. After several years my faithful terminal has not learned yet that my finger always slips on a "k" and types a "j". Computer software and hardware exhibit much too little flexibility to make them really user friendly, make the man-machine interaction pleasant, and promote a wide diffusion among unexperienced users. It is a pity. Computer technology has reached a degree of complexity suitable for non trivial application in everyday life.

The ultimate goal of the research in learning is to blow life into computers in the form of flexibility and adaptability. We only need to observe primitive life forms to be convinced that adaptive behavior is possible at any level of functional complexity, and the search never ends. This thesis is a contribution to make that goal closer.

A large amount of the work spent in any software agency -academic, commercial, or public- is for software maintenance and updating. This work could be saved in part if it were possible to write evolving, self-updating programs.

In Programming Languages key design criteria are writability, readability and reliability. The picture is clear: we want them to look like natural languages. Moreover when a mechanism developed by research in computational linguistics is reliable enough, it can be employed in fields like Programming language where reliability is essential. Natural language is believed to be the major medium by which a culture bequeaths its achievements, what it has learned. The knowledge of the laws which govern this mechanism are likely to be of interest to both Programming Language and Computational Linguistics.

Transmission of information, communication, is a key element present in many research fields. It interests computers technology at every level. The Von Neumann

style computer is a device that moves around data and performs simple operations; then information is moved through several levels of memory and peripherals. The increasing power of computer systems has made it possible to network systems all around the world. In general, theoretical and practical issues in parallel computing are concerned with making the information that a processor has reached a certain state, available to other processors.

The importance of efficient, reliable communication, and standardization is vital to all these fields. They will benefit much of the data about the means of communication that mankind has been perfecting over the past few tens of thousand years. It is of great interest to computer science and technology to gain knowledge about the laws by which information is stored in language and about the reason why languages defined *ad hoc* like Esperanto are difficult to use and not as efficient as languages that evolved naturally [84].

## 1.2 This Thesis

Before trying to explain what was achieved in the present thesis, we need to consider what is meant by the term"learning" and why the concept is often used in close connection with "inductive inference".

The relations between learning and inductive inference have been effectively pointed out by Angluin and Smith [1]. The common sense concept of "learning" is "to gain knowledge" or, in other words, to acquire data. "Inductive inference" denotes the process of hypothesizing a general rule from examples.

### Learning as Efficient Acquisition of Data

It is clear then how inductive inference is useful for learning: a general rule is usually a more efficient representation for a corpus of data. We get to our first characterization of the concept of "learning" as considered in this dissertation: *acquiring data in an efficient way.* What "acquiring data" means should be sufficiently clear to a computer scientist, however "in an efficient way" requires some further attention. Part of this thesis contributes to a clarification of this idea by exploiting

basic concepts from Information Theory where, for several decades, researchers have studied the efficiency of data transmission.

Our ultimate aim is to provide the guidelines for the construction of a learning system that obeys laws of spontaneous convergence towards more efficient operation. We start our task at the level of perception: the system transforms the sensory data into more efficient representations. The short term construction of complex representation is *understanding*. The long term construction of the means to obtain a complex representation is the process of *learning*.

Efficiency is defined as *minimal cost* where *cost* is any appropriate function. Our exposition does not depend on a particular choice of the cost function (except that it has to satisfy natural monotonicity constraints); it lends itself to many different interpretations. A suitable cost function is *size*, memory space; in this situation "efficient" is synonymous to "short". The process of finding a more efficient representation can be justified with the necessity to fit a larger and larger amount of data in a small memory space. Another appropriate cost function is *time*; "efficient" is synonymous to "quick", and finding a more efficient representation can be justified with the necessity for real time computation.

### Learning as the Search for Structural Descriptions

Nevertheless we emphasize that, while learning, one acquires data with the purpose of using it in the future, *in toto* or in chunks. Supposedly a corpus of examples contains some relevant pieces of information that one would like to extract. Our goal is not only a compressing algorithm, like Ziv and Lempel's [86] which already shows many optimality characteristics; what we desire is a way *to find a structural description of the data*. We get to the second point covered by this thesis: what is "structure" and what are the operations needed to discover it.

We will discover two operations to make a representation more efficient: block coding and class coding. While block coding is a textbook technique in information theory, we are not aware of any previous proposal of class coding as an optimization technique. Their description is in this thesis is kept to the bare essential, but they are the object of two related research fields: Pattern Recognition and Pattern

Classification.

By means of these two operations we will be able to construct structures in the form of what we call *interpretational schemes in a context*, or *alphabets*. We will find strong evidence that this formalization of structure is as expressive as other established formalizations, e.g. grammars, programs or Turing machines.

### Learning as the Construction of a Semantics

The great interest that computational linguists show for the parse tree of sentences is due to the belief that with it they can retrieve the meaning of the sentence. This is true in general: structural description and meaning are closely related. We are led to consider a third definition of learning: *constructing a semantics for the data under observation.* This definition is a little more difficult to grasp, especially because "meaning" and "semantics" are not yet completely understood concepts. We can get an informal intuition of the idea: when children learn their first language they attach a meaning to a few strings of phonemes and incrementally proceed to discover the meaning of longer and longer utterances. In this dissertation there is an attempt to formalize the idea of "semantics" and hence to clarify what the phrase "to construct a semantics" means.

The embryonic results achieved are sufficiently clear to give a definite feeling for what the semantics is, how it is in relation with reference and in what way the semantics of a symbol depends on its context. We hope that the formalization proposed will why the construction of a structural description is in fact the same as the construction of a semantics and will capture the reader's common sense idea.

### Learning as the Construction of an Efficient Language

Children, when learning their mother tongue, construct their own language, which should coincide with the language of their community. However, the language constructed is one's own: there are cases of twins who develop a language which is not that of their community. We have pointed out that learning has many relations with inductive inference and the theory of inductive inference hinges upon the notion of *language*, as we will see in the historic introduction to the problem. Languages

are in one to one correspondence with classes of grammars, or classes of any other computational devices like programs, Turing Machines, etc. Informally speaking, as a result of this one to one correspondence, we can identify the notion of language with that of computational device. If we take the learning set to be a text, then a problem of inductive inference becomes that of constructing a language for that text (we allow pluralism and will not say "the language of that text"). Then, learning becomes *constructing an efficient language for a given text.*

The development of this point reduces to pointing out equivalences between well established notions. For this reason we will ground it on experiments with real linguistic data and on the exposition of examples of the concepts constructed by other known ideas that they want to model.

This dissertation will clarify why the various definitions for learning given above do, in fact, coincide. The knowledge that acquiring data efficiently, finding structural descriptions, constructing a semantics, constructing a language, all describe "learning", can be very useful in finding relations and applications in many different fields of science. For instance, in Section 3.3 we will consider an attempt to bridge the gap between the so called "symbolic" and "subsymbolic" learning. Our belief is that the difference is only in the way of expressing the ideas and that there exists a thorough coincidence in the underlying mechanism.

## 1.3 A Bird's Eye View

The present chapter proceeds with a historical introduction to problems in different fields addressed by this thesis, so as to point to possible applications of the results achieved. However, no knowledge in those fields is required as prerequisite for the exposition in later chapters. The theory presented is self-contained. Some examples, however, might require further external information.

The introductory chapter is followed by three main chapters. Also these smaller subunits are reasonably self-contained: they don't depend on each other except for some definitions and notational conventions and they can be read independently. However the thesis will be fully appreciated only when the true connections between

the three main chapters are drawn.

Chapter 2 is devoted to the construction of a rigorous language so as to give a concrete meaning to the several working definitions for learning given in the statement of our purpose. All the expressions used will be given a precise formalization on the basis of few elementary notions.

The mathematical approach that we embraced in introducing our new language gives another advantage besides rigor: the terms involved should be taken with no preconceived interpretation but only in their explicit mathematical relations. The reader is only supposed to understand, not also to believe. Nontheless we will use familiar names to favor the interpretation that helps in understanding.

The entire dissertation hinges upon the notion of *alphabet*, so the main aim of Chapter 2 is to provide a rather general definition of alphabet and to show that it is grounded on the notions of *code* and *classification*. Nonetheless some results presented are interesting in themselves. The first is Theorem 1 which governs the proper interaction between patterns and classifications. The second is Theorem 2 which shows that the structure constructed is as powerful as any other computational structure.

As we have pointed out, it is not mandatory to read Chapter 2 in order to understand the following Chapter 3 and Chapter 4. The reader can do without it and take *alphabet* as the generalized notion of the familiar concept as it is done in the theory of free semigroups.

Chapter 3 deals with the notion of measurement which is a fundamental concept for any system trying to gain knowledge. Once again ideas will be studied in their full generality. The main aim of this chapter is to clarify the notion of *efficiency* with respect to a given cost function. In doing so we will be able to recognize that there exists a gain in efficiency when defining classification but only if the classification is considered jointly with a set of patterns, i.e. a code. In this chapter we will draw connections between the ideas constructed with the notions of energy, entropy and temperature and we will be able to look at learning as a process of evolution.

Chapter 4 is devoted to pragmatics and will contain general learning algorithms based on the notions of previous chapters. We will introduce the technique of hi-

erarchical search in order to improve the running time of the obvious enumerative algorithm. The main idea set forth points to the necessity to introduce cost functions, like running time and accuracy, as parameters to the learning algorithms constructed. We will build up algorithms which are feasibly implementable.

## 1.4 Historical Motivations

There is always a temptation to explain a general idea without giving any example, in order not to freeze the reader's imagination on a particular theme. We will overcome this temptation and ground the exposition on the explanation of the problems and the results that, historically, have motivated much research in the field. The following sections provide the necessary points of connection with the fields of inductive inference, algorithmic information theory, pattern classification, hierarchical systems, and language acquisition.

### 1.4.1 Inductive Inference

#### Enumeration

The first step in many scientific enterprises is to translate its problems into formalized language. Our goal is to construct a mechanism which provides a structural description for the reality under observation and these concepts have been formalized in many equivalent ways. We adopt the one exposed in the following.

The reality for which we want to find a structural description (or still better, a general rule) is presented as a sequence of symbols, i.e. in the form of a text $\mathcal{T}$. Let us assign a number to each symbol, then the text becomes a sequence of numbers. Now we have to make a distinction and decide whether we want a general rule for the sequence as an ordered set or simply as an unordered set, in other words, we have to decide whether the sequence is one big instance or is a set of examples. This distinction might bother us if we think back to the general problem, but, under our model, the two different cases can be treated essentially in the same way, with minor

differences in details.

With no loss of generality (at least in principle) we can formulate the problem in a more circumscribed domain of recursive function theory as it has been done by Gold [38]and Blum and Blum [6]. The sequence is a recursively enumerable set and the description for that set is a partial recursive function. The solution was given in [6]: for several, quite general classes of recursive enumerable sets it is possible to construct an algorithm which yields such a partial recursive function by looking at one element at a time. The algorithm is based on the enumeration technique whose general idea runs as follows: enumerate all the partial recursive functions and check if they halt on all the elements seen so far. Eventually the right one is hit. Of course things don't always go smoothly, one needs a little care in dovetailing and in considering only partial recursive functions for which the predicate $[\phi_{\sigma(i)}(x) = y]$ is decidable. See [6] Example 2.

It is now easy to stretch the concepts and take the description of a sequence -any kind of symbols- to be any computational model that generates that sequence as an output. Then the enumeration technique can be applied also to different domains like grammatical inference, as it has been used by [5] whose system proceeds to construct a grammar by successively adding rules that are consistent with the positive examples observed.

Unfortunately the framework of identification by enumeration raises two sorts of forethoughts. The first follows the observation that there are, in general, many partial recursive functions which identify the same set as well as many grammars which identify the same language. The second concerns the high computational cost of an enumeration.

### Information Complexity

Recursive functions, grammars, or programs as described above, are *representations* for the given text. We have pointed out that we aim at efficient representation[1] so we also need to formalize notion of "best". The "best" description for a text

[1]This is obviously just a standpoint motivated by the trend common to much scientific work and discovered in much natural events summarized by Whitehead's *law of universal lazyness*

(among all the possible ones) is achieved by considering what is "optimal with respect to a given cost function". So if the cost function is *length*, then the best description will be the one whose length is the same as Kolmogorov complexity[2] $K$ of the sequence. Unfortunately Kolmogorov complexity is not computable [87] and that creates a problem in using the concept of *minimal description* [52,70,71]. One of the motivations for the present thesis is the necessity to use another well established information complexity, Shannon entropy which bounds $K$ from above ([86] Theorem 5.1). Shannon entropy, or any other entropy-like complexity [18], is the natural complexity measure for the *alphabet*, the model of structure used in this thesis.

The relations between Kolmogorov complexity and Shannon entropy have been studied from several standpoints [19,20,29,31,55,87]. We found the use of Shannon entropy interesting because our situation allows one to change the scheme with respect to which complexity is computed. Relative entropy is more easily computed in this way because we don't need to go through the universal scheme.

### Continuity

In order to understand why the enumerative solution presented above is not effectively workable we should make the following observation. For the sake of our reasoning let us choose programs as computational model; analogous considerations can be made with any other different choice. Enumeration of programs is based on the choice of a conventional Gödel numbering [30] which is constructed by considering the exterior appearance of the program. *It is an intrinsic property of Gödel numbering that programs which have a similar behavior and similar exterior appearance can be mapped to distant numbers.* Let us try to illustrate this phenomenon with a different example. Rational numbers are denumerable but there exists no enumeration that preserves their natural order. So, if one is asked to find a rational within a given distance $\varepsilon$ from a given irrational number, and, as only means of producing numbers, one is given an enumeration of the rationals, then he will never

[2]Kolmogorov complexity is defined just as the length of the shortest program that computes the sequence with respect to a universal computing scheme [51,87,19,20]. A sequence is *structured* if its Kolmogorov complexity is smaller than its length.

be able to use the information that a particular choice was close. He will have to stick with the enumeration and wait.

This thesis is an attempt to render the representation of the computational model, of its input, and of its output, close enough in appearance so that a small change in the input would cause a small change in the output for a given instance of the model (program), and such that a small change in the program would cause a small change in the output for a given input. This result is achieved with the introduction of the *alphabet* as a computational model. However, we should emphasize that the present work is just a first step towards the goal of identifying the intuitive notion of continuity as explained above. Its rigorous understanding looks by far a more difficult task in our irredeemably discrete domain. This problem has also been addressed in the different domain of lambda calculus and it is discussed in [4].

### Computational Complexity

The model that we introduce allows one to change the enumeration once he has got close to the desired result so as to adjust the range of variability of the enumerative process and acquire efficiency in computation. As we will see this brings about a qualitative speedup of the convergence toward the solution. Our approach is close to that of Solomonov [81] and Rissanen [69,70,71,72] who suggested that the index of the enumeration can be combined with a measure of the complexity of the sequence to get a good description. The same intuition has been used also by De Santis *et al.* [33] for probabilistic prediction functions. The measure selected by our approach is the entropy of the text relative to the description.

The delicate point of high computational cost involved with many solutions of learning problems has been addressed most properly by Valiant [83] with a definition of what is *learnable* in terms of what is achievable with an algorithm which runs in time polynomial in the accuracy parameter $h$ and in the various size parameters of the program to be learned. Our model is quite different from the one in [83] but it is not difficult to find the right relations. The algorithms that we present do have a parameter which corresponds to accuracy.

### 1.4.2 Pattern Recognition and Pattern Classification

Pattern recognition and pattern classification are probably the oldest fields of research closely related in problems of learning. The idea of pattern is used throughout the present work in its limited formulation as *ordered n-tuple of symbols*, hence synonymously with string.

Many different classifier systems exist which employ different standpoints. See [35] for a good introduction. Classifiers are systems that try to sort a set of instances according a preconceived set of features.

This thesis addresses the following two problems, concerning patterns and classes in a fundamental way:

1. We are trying to model learning so in some way we need to do both classification and recognition of characteristic patterns. So, what are the good patterns and what are the good classifications to look for, i.e. to construct, in order to have a productive interaction between the two? In other words, what are the conditions that sets of patterns and classifications must satisfy so that one can effectively apply these two operations again to get a further improvement?

2. How does classification fit in our scheme of learning as efficient memorization?

Theorem 1 gives an answer to the first question. The second question is answered in Section 3.1.3.

### 1.4.3 Hierarchical Systems

The idea to construct hierarchies of alphabets with the goal of inductive analysis of a text was first employed in [9,10][3] which also provide a procedure, the *Procrustes* algorithm, for constructing a higher level code for the text. A similar standpoint lead to the *excision* procedure [42] independently used by the Linguistic String Project [74,39] for constructing their natural language grammar.

[3] In this dissertation we use the concept of alphabet in a more general way. Our *code* is what in [9] is called alphabet.

The Procrustes vein of research lead to the conception of hierarchical systems [8] as optimizing self-organizing systems. A great deal of experimentation [11,65] has shown that the theory of hierarchical systems describes the behavior of many natural system, like monetary systems, human population systems, and natural languages.

The results with natural language give the heuristic evidence that the choice of where to draw level subdivisions is arbitrary. This evidence has motivated the present research for a general homogeneous approach to all levels of language, on the guidelines of [9]. A similar standpoint is taken by stratification linguistics: in [58] we can find evidence for the claim otherwisely supported in this thesis (See Theorem 2) that a lot of linguistic facts can be expressed with structures constructed with strings and sets.

### 1.4.4 Language Acquisition and Natural Language Processing

This section provides a more detailed introduction to the fields where the results in this thesis find immediate pragmatic application.

Our problem in this domain is language acquisition. It can be formulated in the following way: how to build a device that learns the language it is exposed to, and what is the *a priori* machinery that is needed to do this. This problem, with this simple formulation, has brought about endless discussion. [67,68] give a good glimpse on it.

Discussion is often caused by misunderstanding or, better, different understandings of the arguments involved. In the given formulation of the problem there are many terms which allow multiple interpretations, but we don't want to get into philosophical arguments. Let us simply suppose that the aim is to construct a device which will acquire knowledge about language in the form that is needed by a natural language processing system. So we need to know what a language processing system is. The following general description of a language processing system is intended for the nonspecialist. The expert should bear with its oversimplifications.

<S> ⟶ <SUBJ> <VERB> <OBJ>
<SUBJ> ⟶ <NSTG>
<PN> ⟶ *P* <NSTG>
<NSTG> ⟶ <LNR>
<LNR> ⟶ <LN> *N* <RN>
<LN> ⟶ <TPOS> <APOS>
<TPOS> ⟶ *T* | *null*
<APOS> ⟶ *ADJ* | *null*
<RN> ⟶ <PN> | *null*
<VERB> ⟶ <LTVR>
<LTVR> ⟶ <LV> *TV* <RV>
<LV> ⟶ *D* | *null*
<RV> ⟶ *D* | <PN> | *null*
<LVR> ⟶ <LV> *V* <RV>
<OBJ> ⟶ <NSTG> | <TOVO> | *null*
<TOVO> ⟶ *to* <LVR> <OBJ>

Figure 1.1: A Natural Language Grammar

Most current systems consist of three parts: a syntactic, a semantic, and a discourse analyzer[4]. This subdivision already gives an important piece of information: syntax, semantic, and discourse analysis are believed to rely mainly on different mechanisms.

### Syntax Analysis

Syntax analysis is performed with the use of a grammar and a dictionary emebedded in the system. An example of a natural language grammar is given in Figure 1.1 which is a simplified version of the one given in [39].

Symbols outside of angular brackets, the terminals of the grammar, are word classes and should be read as

*ADJ*: adjective *D*: adverb *N*: noun

[4]This is in fact the chapter organization of [39]

$P$: preposition $T$: article $TV$: tensed verb
$V$: untensed verb

The following is an example of a dictionary.

blue: $ADJ$
cheese: $N$
John: $N$
like: $ADJ$, $P$, $TV$, $V$
likes: $TV$
Mary: $N$
with: $P$

When the system is presented with a sentence, say "Mary likes blue cheese", the system searches the dictionary and finds the word class attached to each word in the sentence. Then it replaces each word by its syntactic class and tries to parse the sentence with the grammar by constructing a parse tree. Note that there might be more then one choice available in several steps of the process. This leads to the problem of finding the "right" choices or, in other words, the right parse tree. The solution is usually pursued in the direction of strengthening the grammar. It can be achieved by imposing additional constraints, like agreement in number between verb and subject and agreement in attributes found on the dictionary. However, in principle, strengthening the grammar can always be achieved with additional grammar rules.

We can now ask a few questions which lead to the arguments that motivated this research. First of all, why is the dictionary not incorporated in the grammar? In fact, one could decide that word classes are variables, and dictionary entries terminals of the grammar. Second, many of the *restrictions* [39] (the constraints that guide the applicability of a rule) concern situations that are usually indicated in the suffixes of the words, like number or tense. So, why is the dictionary, i.e. the set of the terminal symbols, not pushed to the higher resolutions of roots and

suffixes, or even to the level of letters? The previous two questions can be rephrased in the following way: what guided the apparently arbitrary choice of the level at which to freeze the terminals of the grammar?

The best reply to these inquiries is that, though arbitrary, the choice was dictated by convenience. For historical reasons, information about the dictionary the way it was chosen was readily available with particularly interesting semantic value. Moreover, choosing the terminals at a lower level would probably involve a higher computational complexity of the parsing procedure. Least but not last, most applications are concerned with written language where there exists a special symbol, namely the space, which indicates the end of a word: there is a natural ending.

This answer leads to a new question: What are the mechanisms that guided the history of research in mathematical and computational linguistics to concentrate on this level? Is it just because that is about the right level of computational complexity?

Obviously much information is lost by drawing this apparently arbitrary level line; so most serious systems try to integrate the other levels. Is there a way to treat all the levels homogeneously?

### Semantic Analysis

The second step that a natural language processing system takes is semantic analysis. It aims at attaching a meaning to the tree structure constructed during the syntax analysis. Stated in this way, the problem looks hopeless. What is meaning, where is it located, and how can we retrieve it from the structural description are difficult questions which are not very well understood. For this reason researchers prefer to describe it as the translation of the sentence into a formal language which should "be unambiguous, have simple rules of interpretation and inference, and in particular have a logical structure determined by the form of the sentence." [39]

The formal language usually chosen is that of logic. The side-effect to its nice properties quoted earlier is that it imposes a certain literal-mindedness to the system. Often the literal interpretation of an utterance has very little to do with the intended one. This considerations, in fact, led to a vein of research [2,27] stemming

from the *Speech Acts* approach to philosophy of language [3,76]. Is that the vein of contradiction asking the question: Why should we translate a sentence into logical form and not treat it the same way human beings treat it? Do they treat it with logic?

A final answer to these questions requires an adequte theory of language understanding. Unfortunately we still miss one. In this situation one cannot help looking for a solution to the difficult questions concerning meaning.

In this thesis we will embrace the idea that meaning is global, distributed property of a given expression. We call this *the holographic hypothesis.* It can be exemplified in many ways by considering obvious characteristics of language like redundancy. Consider for example "W r ntrstd n rdndnc". The same trick can be played at every level of language, not only that of letters. It can be done with syllables, words, phrases, sentences, periods, ..., provided the context is appropiately enlarged.

### Discourse Analysis

We have introduced enough redundancy so that there is no need for the explanation that discourse analysis is the attempt to find the interconnections between different sentences in a text, once again, to retrieve its meaning. However, we can't help asking the same question: why is it considered different than the previous steps? Why are the tools and the theories that are used different [5]?

Let us go back to our language acquisition problem. The aim is to construct a device which will acquire knowledge in the form that is needed by a natural language processing system. As we have seen, currently it is composed by quite differentiated subsystems, so the language acquisition task is different depending on what subsystem one is thinking about. As a matter of fact, different works on language acquisition, starting from different assumptions, use different methods and achieve different goals. As an example consider [5] and [77]; both claim to address the language acquisition problem but their results and methods are very far apart.

[5] Actually there have been indications that discourse analysis can be achieved with the same means of syntax analysis with the *text grammar* approach but they are not very effective and other methods are more fashionable

This thesis is based on the understanding that the various steps of natural language processing -as is currently approached- are instances of the same problem of *finding the interconnecting relations between sub-units that form larger constructs.* The following are different instantiations of the previous sentence obtained by assigning more specific words to the general concepts *units* and *bigger constructs.*

1. find the interconnecting relations between words in a sentence;
2. find the interconnecting relations between word meanings forming a sentence meaning;
3. find the interconnecting relations between sentences in a discourse

The general problem of language acquisition becomes that of constructing the machinery for obtaining these relations. In general, the problem of learning is that of constructing a machinery that permits the discovery of the relations between sub-units in a larger structure. The present thesis gives a precise formulation and a workable solution to the general problem.

# Chapter 2

# Semiotics

The language introduced in this chapter aims at describing the concepts and the situations which are encountered in semiotics, in general, and linguistics, in particular. We define the important notions of semantics, syntax, interpretational scheme and context in such a way that the definitions emphasize their interrelations. Then, we introduce two structure constructors to point out the connection between structure and semantics. We will see how structure permits representation and how the two constructors interact to create a unique structural concept, the alphabet.

We introduce a computational paradigm and we prove that it is as powerful as any other established computational paradigms. This gives evidence to the fact that the two constructors are sufficient to create any computable structure.

The proposed theory will be rigorous only up to the point when its structure is solid enough to compare the notions involved with other related ideas. We will see that the intuition that we have built is in harmony with other different characterizations for the concepts under consideration. It is not by chance that rigor is synonymous with stiffness; therefore the use of the rule to define and work out everything is not employed to the extreme. The more we proceed in the study, the easier will be to introduce new concepts and statements. We will avoid all the tedious details when they could be easily reconstructed on the base of the context and of previous assertions. "The last dotting of the last i, in the manner of the old fashioned Cours d'Analyse in general and Bourbaki in particular, gives satisfaction to the author who understand it anyway (...); for most serious-minded readers it is

worse than useless." [41].

## 2.1 Symbols and Symbolic Forms

The key idea is once again to split in halves, to impose a distinction when apparently there is not; the concepts that we separate are "sign" and "symbol". We call signs *symbolic forms*, in order to free the reader's imagination in interpreting. Symbols and symbolic forms are the elementary notions of the theory. Unless otherwise stated, the universe of symbols will be disjoint from the universe of symbolic forms.

Let $\mathcal{U}$ be a universe of symbols and $\mathcal{L}$ a universe of symbolic forms. A *constructor* $\omega$ is a function $\omega : \mathcal{U}^* \longrightarrow \mathcal{L}$, where $\mathcal{U}^*$ is the universal language over $\mathcal{U}$, i.e., the free semigroup over $\mathcal{U}$.

Given a set of symbols $\mathcal{U}$ and a family $\Omega$ of constructors, we can conceive of a symbolic form over $\mathcal{U}$ as the result of the application of an $\omega \epsilon \Omega$ on arguments in $\mathcal{U}$. Let $\mathcal{L}_n(\mathcal{U}) = \{\omega(s_1, ..., s_n) | \omega \epsilon \Omega, s_i \epsilon \mathcal{U}\}$ and $\mathcal{L}^+(\mathcal{U}) = \bigcup_{i=1}^{\infty} \mathcal{L}_i(\mathcal{U})$. $\mathcal{L}^+(\mathcal{U})$ is the set of all the possible symbolic forms on the symbols of $\mathcal{U}$ and will be called the *hyperuranios* of $\mathcal{U}$.

Constructors should be thought of as all the functions that allow the formation of complex symbolic structures from an ordered set of symbols. This justifies the name hyperuranios; with it Plato denoted the space beyond the celestial spheres where Ideas reside.

**Example 1** 1. Let $\Omega$ contain only the string constructor [.], then $\mathcal{L}^+(\mathcal{U})$ is the free concatenation (or the universal language) over $\mathcal{U}$, $\mathcal{L}^+(\mathcal{U}) = \mathcal{U}^*$.

2. Let $\Omega$ contain only the set constructor $\{.\}$, then $\mathcal{L}^+(\mathcal{U})$ is the powerset of $\mathcal{U}$, $\mathcal{L}^+(\mathcal{U}) = \mathcal{P}(\mathcal{U})$.

□

**Example 2** (i) Let $\mathcal{U}$ contain the symbol $\emptyset$ and $\Omega$ the only element $\omega$. Let us suppose that $\omega$ is injective. Then $\mathcal{L}^+(\mathcal{U})$ contains infinitely many symbolic forms,

namely $\omega(\emptyset), \omega(\emptyset, \emptyset), ....$ Under these hypothesis we cannot say more about $\mathcal{U}$ or $\mathcal{L}^+(\mathcal{U})$.

**(ii)** However, if we make the additional supposition, as many mathematicians do, that $\mathcal{L}^+(\mathcal{U}) \subseteq \mathcal{U}$ then we can easily prove that $\mathcal{U}$ as well contains at least countably infinite elements.

□

### 2.1.1 The Semantics

We have assumed that the $\omega$'s take only symbols as arguments, so if we want to apply them recursively on new symbolic forms created, we have to associate a symbol $s \epsilon \mathcal{U}$ to every symbolic form $\omega(s_1, s_2, ...), s_i \epsilon \mathcal{U}$ by means of a map $\phi$. In this way any symbolic form can serve as a symbol itself.

$\phi$ will be given the name *atomizing function* to suggest the idea that it is a function that allows to look at a complex symbolic construction as one single atomic symbol.

The alternative name *semantics* [1] is justified by the *holographic hypothesis*; meaning is a global property of a given expression.

**Definition 1** An *atomizing function* for $\mathcal{L}^+(\mathcal{U})$ or simply a *semantics* is a map

$$\phi : \mathcal{L}^+(\mathcal{U}) \longrightarrow \mathcal{U}$$

satisfying the property:

$\phi 1)$ $\quad \phi(\omega(s)) = s$ for each $\omega \epsilon \Omega, s \epsilon \mathcal{U}$

The semantics is *faithful* for $\Omega$ if it is one to one (except, obviously, for elements in $\mathcal{L}_1(\mathcal{U})$: by property $\phi 1$ it can be one to one only if $\Omega$ is a singleton set). The triple $(\mathcal{U}, \Omega, \phi)$ is a *semantic universe.*

□

[1] The etymology of the greek word *sēma*, sign, symbol, roots it in the Indo-European *dhyāmn* and the Sanskrit *dhyāna*, thought.

Property $\phi 1$ is quite natural and corresponds to the idea that by using only one symbol it is not possible to express, or write down, anything different than the symbol itself.

We will be operating only on the universe of symbols. Therefore, no matter what the operations in $\Omega$ do, our only concern is what the semantics returns in $\mathcal{U}$. Therefore a faithful semantics is one which reflects exactly the behavior of the $\omega$'s. Moreover, the hyperuranios of a universe is always constructed by means of the class $\Omega$, so we can interpret the semantics $\phi$ as the semantics of the operations in $\Omega$. By means of $\phi$ we can see the $\omega$'s in $\Omega$ as operations on $\mathcal{U}$, $\omega \circ \phi : \mathcal{U}^* \to \mathcal{U}$.

The following examples should clarify the matter.

**Example 3** Let $\mathcal{U}$ contain (at least) two symbols, $0, 1 \epsilon \mathcal{U}$ and $\omega \epsilon \Omega$. Then $\omega$ is the operation of set formation if

$$\phi(\omega(s_1, ..., s_n)) = \phi(\omega(s'_1, ..., s'_m)) \Leftrightarrow \{s_1, ..., s_n\} = \{s'_1, ..., s'_m\} \qquad (2.1)$$

If we identify the set $\{0, 1, ..., n-1\}$ with the number $n$, $\mathcal{U}$ contains all the natural numbers and (almost all) the finite subsets of the natural numbers.

Let us emphasize that condition $\phi 1$ makes sure that the set containing only one symbol is identified with the symbol itself. We assumed condition 2.1 in the situation of Example 2 (i), still we could not say anything more than $\emptyset \epsilon \mathcal{U}$. The qualitative jump is given by the presence of two elements in $\mathcal{U}$.

On the other hand, as is well known, if we dropped condition $\phi 1$ and allow to see the set containing only one element as distinct from the element itself, we will fall in the same situation as Example 2 (ii) where the hyperuranios is contained in the universe.

□

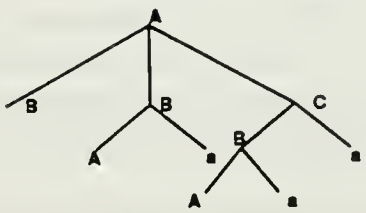
**Example 4** A context-free grammar with set of variables $\mathcal{V}$ and set of terminals $\mathcal{W}$ defines a semantic universe supported by the universe $\mathcal{U} = \mathcal{V} \cup \mathcal{WS} \cup \{\uparrow\}$. The special symbol $\uparrow$ is read as *meaningless*; $\mathcal{S}$ is a set of additional symbols.

In order to see it we need the following

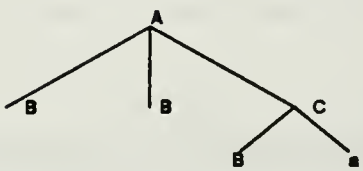
**Definition 2** A *pruned tree* of a tree $t$ is a tree obtained from $t$ by pruning off some subtrees but leaving their roots.

□

**Example 5** .




*A tree t*




*A pruned tree of t*

□

Consider a context free grammar $G$. Let $\Omega$ contain only the operations of string formation and set formation.

For all the elements $s \in \mathcal{L}^+(\mathcal{U})$ define $\phi(s)$ as follows:

- if $s$ is composed of a single symbol then $\phi(s) = s$;
- if $s$ is a string and can be read off the leaves of some pruned subtree $u_1, ..., u_n$ of some derivation trees $t_1, ..., t_m$ of $G$, then put the additional new symbols

$w_{u_1}, ..., w_{u_n}, w$ in $\mathcal{S}$. Let $\phi(w_{u_i}) = root(u_i)$ and $\phi(\{w_{u_1}, ..., w_{u_n}\}) = w$, $\phi(s) = w$

- otherwise $\phi(s) = \uparrow$

□

**Example 6** Let $(\mathcal{U}, \Omega, \phi)$ be the semantic universe associated as in Example 4 to the grammar $G$ for natural language given in Section 1.4.4. Denote with $\mathcal{Z}$ the set {SUBJ, OBJ, VERB} and with $\mathcal{S}$ the set {S}. Then

1. The semantics of SUBJ VERB OBJ is S;
2. The semantics of $T$ $N$ is NSTG;
3. The semantics of $TV$ is $TV$.

□

The are many possible ways to show that model theoretic semantics is captured by our definition. In the following example we will work out the case of Propositional Calculus and Predicate Calculus.

**Example 7** As a first step let us consider Propositional Calculus; we will see how to construct a semantic universe from a set $V$ of propositional variables. Let $\mathcal{U} = V \cup \{\Rightarrow, F, T, \uparrow\}$, $\Omega$ contain only the string formation. In this situation $\mathcal{L}^+(\mathcal{U}) = \mathcal{U}^*$. A model theoritic semantics $\phi$ is a truth value assignment consistent with the obvious laws of logic[2]. Ill formed formulas are assigned the value $\uparrow$. So $\phi : \mathcal{L}^+(\mathcal{U}) \rightarrow \{F, T, \uparrow\} \subseteq \mathcal{U}$

Let us now consider Predicate Calculus. Given a set $C$ of constants, a set $P$ of predicates, a set $V$ of variables we define $\mathcal{U} = C \cup P \cup V \cup \{F, T, \Rightarrow, \forall, \uparrow\}$, $\Omega$, and $\phi$ as above.

□

[2]The set $\{\Rightarrow, F, T\}$ is just a possible minimal set of symbols for Propositional Calculus.

### 2.1.2 The Syntax

A widespread belief that motivated this research is that syntax and semantics can be classified as different things. This belief is somewhat misleading. Syntax and semantics are different, but they also bear a lot of connections: in this section we define syntax as the inverse function of semantics. This makes it possible to understand why learning can be explained both as the construction of a semantics and as the construction of a structural description.

Let $(\mathcal{U}, \Omega, \phi)$ be a given semantic universe. Consider the equivalence relation $\iota$ on $\mathcal{L}^+(\mathcal{U})$:

$$x \; \iota \; y \Leftrightarrow \phi(x) = \phi(y) \tag{2.2}$$

and denote with $\mathcal{L}(\mathcal{U})$ the quotient space

$$\mathcal{L}(\mathcal{U}) = \mathcal{L}^+(\mathcal{U})/\iota$$

Then there exists a unique bijective map $\overline{\phi}$ such that the following diagram is commutative.

$$\begin{array}{ccc} & \mathcal{U} & \\ {\scriptstyle\phi}\nearrow & & \nwarrow{\scriptstyle\overline{\phi}} \\ \mathcal{L}^+(\mathcal{U}) & \xrightarrow[\text{nat}\,\iota]{} & \mathcal{L}(\mathcal{U}) \end{array}$$

**Definition 3** Let $(\mathcal{U}, \Omega, \phi)$ be a semantic universe. Since $\overline{\phi}$ is a bijection then we can consider bijective map $\psi = \overline{\phi}^{-1}$, $\psi : \mathcal{U} \hookrightarrow \mathcal{L}(\mathcal{U})$. $\psi$ is the *reference function*, or simply the *syntax* of $\mathcal{U}$.

Let $\Sigma \subseteq \mathcal{U}$. The function $\psi_\Sigma : \mathcal{U} \to \mathcal{L}(\Sigma)$ defined as $\psi_\Sigma(s) = \psi(s) \cap \mathcal{L}(\Sigma)$ is the *reference function of* $\mathcal{U}$ *restricted to* $\Sigma$.

Let $\Sigma_0, \Sigma_1 \subseteq \mathcal{U}$. $\Sigma_0$ is *sufficient* for $\Sigma_1$ if $\psi_{\Sigma_0}(s) \neq \emptyset$ for each $s \epsilon \Sigma_1$

□

**Example 8** Let $(\mathcal{U}, \Omega, \phi)$ be the semantic universe associated as in Example 4 to the grammar $G$ given in Section 1.4.4. Denote with $\mathcal{Z}$ the set $\{\texttt{SUBJ}, \texttt{OBJ}, \texttt{VERB}\}$ and with $\mathcal{S}$ the set $\{\texttt{S}\}$. Then

1. The syntax of NSTG restricted to $\mathcal{W}$ is

$$\psi_{\mathcal{W}}(\texttt{NSTG}) = \{T\ N,\ N\ P\ N,\ T\ ADJ\ N, ...\}$$

2. The syntax of S restricted to $\mathcal{Z}$ is $\psi_{\mathcal{Z}}(\texttt{S}) = \{\texttt{SUBJ VERB OBJ}\}$;

3. The syntax of S restricted to $\mathcal{W}$ is $\psi_{\mathcal{W}}(\texttt{S}) = \{N\ TV,\ N\ TV\ N,\ T\ N\ TV, ...\}$;

4. $\mathcal{W}$ is sufficient for $\mathcal{U}$;

5. $\mathcal{Z}$ is not sufficient for $\mathcal{U}$ but is sufficient for $\mathcal{S}$.

□

### 2.1.3 The Interpretational Scheme

It is a widespread convention to call any set of symbols an *alphabet*[3] . We have seen in the previous section that if a semantic is assigned to our universe then every symbol of an alphabet has attached to it a class of symbolic forms. We will see now how the context permits to select the relevant ones.

**Definition 4** An *interpretational scheme* $A$, or simply a *scheme*, is an ordered pair of alphabets $A = (\overline{X}, X)$ in a semantic universe, $\overline{X}, X \subseteq \mathcal{U}$. $X$ is the *context* of $A$. ($\overline{X}$ might have no relation with $X$ in general. We choose the overline notation because it is suggestive of the operations that we will introduce in the following with which it is consistent).

We will say that $A$ is *over* $X$ and we will call $\Im m(\psi_X)$ the *support set* for $A$.

An element $s \epsilon \overline{X}$ is *atomic* if $\psi_X(s) = s$. A scheme is atomic if all its elements are atomic.

□

[3]We will use the notion of alphabet informally. Later on we will find a way to formalize it in accordance with the commonly used concept.

**Definition 5** A semantics is *faithful* for the scheme $A = (\overline{X}, X)$ if for each $x \epsilon \overline{X}$, $\psi(x)$ contains exactly one element.

Usually all the schemes that we consider live in the given semantic universe. When the semantics is faithful for the scheme considered, we will say that the scheme is *faithful.*

□

**Example 9** Consider the semantic universe associated to the natural language grammar defined in Example 6. Then $\phi$ is faithful for the scheme $(\mathcal{S}, \mathcal{Z})$, but $\phi$ is not faithful for the scheme $(\mathcal{U}, \mathcal{T})$.

The schemes $(\mathcal{T}, \mathcal{T})$ and $(\mathcal{Z}, \mathcal{Z})$ are atomic.

□

**Example 10** Let $X = \{1, 2, 3, ...\}$ be the set of positive natural numbers, $X_1 = \{I, V, X, L, D, M\}$ the set of the roman numerals, and $X_2 = \{0, 1\}$. Let $\Omega$ contain the operation of string formation, then:

$$\begin{array}{cccc} \psi_{X_1}(1) = I & \psi_{X_1}(2) = II & \psi_{X_1}(3) = III & ... \\ \psi_{X_2}(1) = 1 & \psi_{X_2}(2) = 10 & \psi_{X_2}(3) = 11 & ... \\ \psi_{X_1 \cup X_2}(4) = \{IV, 100\} & \psi_{X_1 \cup X_2}(5) = \{V, 101\} & \psi_{X_1 \cup X_2}(6) = \{VI, 110\} & ... \end{array}$$

□

Contexts can be redundant; it might happen, in fact, that $\psi_X(\overline{X}) \subseteq \mathcal{L}(X_0)$ with $X_0 \subset X$. So we might need the function $\mathcal{A}$ which takes an interpretational scheme $A$ as argument and returns the smallest context which $A$ is over:

$$\mathcal{A}(A) = \bigcap_{\psi_X(\overline{X}) \subseteq \mathcal{L}(X')} X'$$

$\mathcal{A}(A)$ is called *the alphabet of A.*

**Remark 1** Often a scheme $A = (\overline{X}, X)$ is identified with the set $\overline{X}$. This entails that we overload the set theoretical symbols $\epsilon, \subseteq, \cup ...$, and their associated concepts.

If $A = (\overline{X}, X)$ is a scheme then $s \epsilon A$ means $s \epsilon \overline{X}$ if $s$ is a symbol and $s \epsilon \psi(\overline{X})$ if $s \epsilon \mathcal{L}(X)$. Analogous situations hold for the other set theoretical symbols.

□

## 2.2 Strings and Sets

We have introduced the notion of semantics and we have seen that it governs the result of the operations in $\Omega$, unless it is faithful. In this section we assume that the semantics behaves as if there were only two operations in $\Omega$, namely the operations of set formation and that of string formation. We emphasize that this is only a point of view and that the same situation can be expressed by saying that $\Omega$ contains only the set-former and string-former and that the semantics is faithful. In fact, in the exposition to follow, we will take the second viewpoint.

In the sequel we will see how the structures constructed with the set-former and the string-former can be used as means of representation. The main result of this section is Theorem 1. It gives the condition under which set formation and string formation interact to create the more general structure of *alphabet*. The notion of alphabet that will be introduced, generalizes the commonly intended concept. In particular it formalizes the idea of level and clarifies the discussion and the questions proposed in Section 1.4.4.

The first question that comes to mind is: is it possible to do anything interesting only with this two constructors? The answer will come in a later section: the strucures constructed in this way are equally expressive as any other computational structure.

Then one might ask why we should be interested in constructing a theory with two constructors, when we already have a very good one, set theory, which is very successful in doing everything only with the set-former. In fact, using the well known Wiener and Kuratosky definition of ordered pair [57], or any other equivalent one, it is possible to recursively define the concept of n-ple or string.

There are many reasons why we introduce this formalism. First the theory based on these two concepts has a particularly appealing intuition common to different

disciplines (see Section 2.3.1). Second the recursive definition of n-ple based on the Wiener-Kuratosky definition of ordered pair has the computational flaw that there are $n$ calls to the first element, $n-1$ calls to the second, ... Since our hypothetical applicative need will mainly be that of recursively determining the way a particular symbol has been constructed, then when we are dealing with a string the process is not parallelizable or, from another point of view, is computationally costly. Third, as we will see the notion of string is fundamental for the further developments of this work in the following chapters. Finally, two is better than one, we are in the situation where we choose a binary over a unary representation[4]. See also Example 3.

### 2.2.1 Codes and Classifications

From now on $\Omega = \{\omega_1, \omega_2\}$ where $\omega_1$ is the constructor of string formation and $\omega_2$ is the constructor of set formation. We adopt the notational convention $\omega_1(s_1, ..., s_n) = [s_1, ..., s_n]$ or simply $s_1 ... s_n$; $\omega_2(s_1, ..., s_n) = \{s_1, ..., s_n\}$. Note that the two constructor take only symbols as arguments, so sets are always flat.

Given a set of symbols $\Sigma$, $\mathcal{P}(\Sigma)$ denotes the powerset of $\Sigma$, $\Sigma^*$ the free semigroup over $\Sigma$. It should be noted that, if we carry out the construction of $\mathcal{L}(\Sigma)$ under any semantics $\phi$, by property $\phi 1$, the string $[s]$ composed of the only element $s$ is identified with the singleton set $\{s\}$.

At this point we need the assumption that the semantics of the universe is faithful for all the scheme that we will consider. Our main interest in schemes is connected to their action on texts, (see following Definitions 8, 9, and 10). We will see in the following that the restriction to faithful semantics does not introduce any loss of generality.

**Remark 2** When the semantics is faithful, we can have a clear picture of what $\mathcal{L}(\Sigma)$ looks like: it contains isomorphic images of $\mathcal{P}(\Sigma)$, $\Sigma^*$, and $\Sigma$ (denoted by abuse of notation with the same symbols). Moreover $\mathcal{L}(\Sigma) = \mathcal{P}(\Sigma) \cup \Sigma^*$ and $\mathcal{P}(\Sigma) \cap \Sigma^* = \Sigma$. □

[4]We should also mention that the designers of SETL [75], the programming language for set theory have chosen to allow two different elementary data structure, the string and the set.

It follows from Remark 2 that a scheme can always be conceived of as the union of two different parts: a code and a classification.

**Definition 6** A *code* is a scheme $S$ such that its support set is a class of strings, i.e. $\psi(S) \subseteq \mathcal{A}(S)^*$

□

A code $S$ is a *formal language* over $\mathcal{A}(S)$ in the common sense, where each word $w$ is assigned the name $\psi^{-1}(w)$.

**Example 11** Consider the semantic universe associated to the natural language grammar as defined in Example 6.

The scheme $(\mathcal{S}, \mathcal{Z})$ is a code. The reader can try as an exercise to find a scheme which is not a code.

□

**Example 12** A more informal but interesting example of code: the set of all procedures with no argument of a compiled Pascal program.

□

**Example 13** The Morse code is as code $\mathcal{M} = (A, M)$ where $A$ is the english alphabet and $M = \{\cdot, -, space\}$. It is assumed that the encoding of every letter ends with a space.

□

In this theory, the counterpart of the code is the classification.

**Definition 7** A *classification* is a scheme $C$ such that its support set is a class of sets, i.e. $\psi(C) \subseteq \mathcal{P}(\mathcal{A}(C))$

□

**Remark 3** A classification $C$ can be thought of as a set of relations on $\mathcal{A}(C)$. If the sets in $\psi(C)$ are pairwise disjoint then C is an equivalence relation or, better,

is the set of the classes of an equivalence relation on $\mathcal{A}(C)$.
□

**Example 14** Consider the set of syntactic classes $\mathcal{W}$ of our natural language example. $\mathcal{W}$ is a classification over the entries of the dictionary: to each element $X \in \mathcal{W}$ is associated the set of all the entries which have $X$ as attribute.
□

**Proposition 1** *Let $A$ be a faithful scheme and $A_0 = \mathcal{A}(A)$.*

**(i)** *There exist a code $S$ and a classification $C$ such that*

1. $\mathcal{A}(S) \subseteq A_0$ *and* $\mathcal{A}(C) \subseteq A_0$
2. $A = S \cup C$
3. $S \cap C$ *is an atomic scheme.*

**(ii)** *There exist a unique code $\tilde{S}$ and a unique classification $\tilde{C}$ which do not contain atomic elements and $A = \tilde{S} \cup \tilde{C} \cup U$ where $U$ is the set of the atomic elements of $A$.*

*Proof:* $(i)$ Let $S = \{x \epsilon A | \psi(x) \epsilon A^*\}$ and $C = \{x \epsilon A | \psi(x) \epsilon \mathcal{P}(A)\}$. 1. and 2. are obvious (see Remark 2). 3. follows from Remark 2 and property $\phi 1$.

$(ii)$ Define $\tilde{S} = S - U$ and $\tilde{C} = C - U$. Existence follow from $(i)$ above. To prove uniqueness we just need to observe that $\overline{S}$ $(\overline{C})$ does not contain atomic elements.
□

**Definition 8** A string $\mathcal{T}$ of elements from an alphabet $A$ is called a *text over $A$*.
□

The goal of the following three sections is to define the action of a scheme over a text. We will construct this definition first by considering how a code and a classification can operate on a text, and then by studying the ways they interact with each other.

### 2.2.2 The Action of a Code and a Classification over a Text

The following two definitions explain how a code and a classification can operate on a text in two different ways. Remember that we assume that our schemes are faithful.

**Definition 9** Let $S$ be a code

1. If $\mathcal{T}$ is a text over the alphabet of $S$, $\mathcal{A}(S)$, then an *encoding* $\overline{\mathcal{T}}^S$ of $\mathcal{T}$ under $S$ is a text over $S$ such that if we replace each symbol $\overline{s}$ in $\overline{\mathcal{T}}^S$ with its associated code word $s = \psi(\overline{s})$ we get $\mathcal{T}$. $S$ is an *uniquely decipherable code* for $\mathcal{T}$, or $\mathcal{T}$ is *uniquely decodable* by $S$ if there exists one and only one encoding $\overline{\mathcal{T}}^S$. $S$ is an *unambiguous code* if it is a uniquely decipherable code for any word of $\psi(S)^*$.

2. If $\mathcal{T}$ is a text over $S$ then the *spelling* $\underline{\mathcal{T}}_S$ of $\mathcal{T}$ under the code $S$ is a text over the alphabet of $S$, $\mathcal{A}(S)$, such that $\mathcal{T}$ is an encoding for it.

□

**Example 15** Let $\mathcal{M}$ be the Morse code as in Example 13. Let

$$\mathcal{T} = -- \quad --- \quad \cdot -\cdot \quad \cdots \cdot$$

then $\overline{\mathcal{T}}^{\mathcal{M}}$ =MORSE

□

**Remark 4** We can generalize, in two different ways, the encoding of a text $\mathcal{T}$ by a code $S$ even if $\mathcal{T}$ is constructed over an alphabet larger than the alphabet of $S$.

1. We assume that all the unknown symbols of $\mathcal{T}$, i.e. the ones which are not in the alphabet of $S$, are deleted beforehand.

2. We assume that the unknown symbols of $\mathcal{T}$ are left unchanged through the operation.

The second case requires a little more attention regarding the convention to take when an unknown symbol appears within a codeword.

Analogous assumption can be made with spelling and the other operations introduced in the following.

□

**Definition 10** Let $C$ be a classification.

1. If $\mathcal{T}$ is a text over the alphabet of $C$ then a *generalization* $\overline{\mathcal{T}}^C$ of $\mathcal{T}$ under the classification $C$ is a text over $C$ such that there exists a way to replace each symbol c of $\overline{\mathcal{T}}^C$ with an element in its associated set $\psi(c)$ so as to get $\mathcal{T}$.

2. If $\mathcal{T}$ is a text over $C$ then a *specialization* $\underline{\mathcal{T}}_C$ of $\mathcal{T}$ under the classification $C$ is a text over the alphabet of $C$ such that $\mathcal{T}$ is a generalization for it.

□

**Remark 5** If $C$ is an equivalence relation over $\mathcal{A}(C)$ then for any text $\mathcal{T}$ over $\mathcal{A}(C)$ there exists one and only one possible generalization under $C$.

□

Encoding and generalization are multivalued unary operations applicable on texts. We will sometime abuse the operational notation and identify their result with only one element in its set of values. Encoding can be undefined if the code does not parse the text. When we use encoding as an operation we will tacitly assume that that is not the case. See also Remark 4.

If $X$ is a code (or a classification) the notation $\overline{\mathcal{T}}^X$ will be read "$\mathcal{T}$ represented in $X$". In general the superscript tells what the text is represented in.

**Example 16** Consider the classification $\mathcal{W}$ of the word classes over the dictionary, constructed as in Example 14. Let $\mathcal{T}$ be the sentence "John likes blue cheese". Then $\mathcal{T}$ represented in $\mathcal{W}$ is $\overline{\mathcal{T}}^{\mathcal{W}} = N\ TV\ ADJ\ N$.

□

**Definition 11** A scheme $A = S \cup C$, where $S$ is a code and $C$ a classification, is *unambiguous* if $S$ is a unambiguous code and $C$ is an equivalence relation over $\mathcal{A}(A)$.
□

### 2.2.3 The interaction of Codes and Classifications

We have just survived a tedious list of definitions and constructions aimed at arriving at the following keypoints:

- An interpretational scheme is composed of two disjoint[5] parts: a classification and a code.
- Classifications and codes independently provide a representation for a given text.

The goal of the present section is to construct in an analogous way the representation of a text given by a scheme. To do this we will look for the conditions under which the code part and the classification part of a scheme interact nicely to give a unique outcome. The findings in this section are summarized in Theorem 1: a scheme provides a unique representation when it is unambiguous and complete.

As a first step towards that goal we need to observe that a text is simply a string so we can immediately broaden the application of the generalization operation to codes. This also allows the definition of the encoding operation on classifications as it is done in the following

**Definition 12** Let $A = S \cup C$ be a scheme, where $S$ is a code and $C$ is an equivalence relation over $\mathcal{A}(A)$.

a) Let $\overline{S}^C = \{\overline{\psi(s)}^C : s \epsilon S\}$. Note that $\overline{S}^C$ is a code over $C$.

b) Denote with $\overline{C}^S$ the equivalence relation over $S$ defined as

$$s\,\overline{C}^S\,t \leftrightarrow \overline{\psi(s)}^C = \overline{\psi(t)}^C$$

[5] In the sense of Proposition 1

□

**Definition 13** Let $A = S \cup C$ a scheme. The *completion* ${}^{C}S$ of $S$ by $C$ is the code with support set

$$U = \bigcup_{s \epsilon \overline{S}^{C}} \underline{s}_{C}$$

$S$ is *complete* with respect to $C$ if ${}^{C}S = S$. The completion of $A$ is the scheme $\overline{A} = {}^{C}S \cup C$. $A$ is complete if $\overline{A} = A$.

□

**Proposition 2** *Let $A = S \cup C$ be a complete scheme ($C$ an equivalence relation and $S$ a code over the alphabet of $A$, $\mathcal{A}(A)$) and $\mathcal{T}$ a text over $\mathcal{A}(A)$ If $\mathcal{T}$ is uniquely decodable by $S$ then $\overline{\mathcal{T}}^{C}$ is uniquely decodable by $\overline{S}^{C}$.*

*Proof:* Let $\mathcal{T} = a_1 ... a_n$ a text over $\mathcal{A}(A)$, $\overline{\mathcal{T}}^{C} = c_1 ... c_n$, and $\sigma = (c_1 : a_1) ... (c_n : a_n)$ the substitution such that $\sigma(\overline{\mathcal{T}}^{C}) = \mathcal{T}$. By contradiction, let $s'_1 ... s'_h$ and $s''_1 ... s''_k$ be two different encodings of $\overline{\mathcal{T}}^{C}$ under $\overline{S}^{C}$. Since $S$ is complete with respect to $C$ then $\sigma(\psi(s'_i)), \sigma(\psi(s''_j)) \epsilon S$ for each $1 \leq i \leq h, 1 \leq j \leq k$. Hence $\psi^{-1}(\sigma(\psi(s'_1))) ... \psi^{-1}(\sigma(\psi(s'_h)))$ and $\psi^{-1}(\sigma(\psi(s''_1))) ... \psi^{-1}(\sigma(\psi(s''_k)))$ are two different encodings for $\mathcal{T}$, a contradiction.

□

**Corollary 1** *Let $A = S \cup C$ be a complete scheme, ($C$ be an equivalence relation and $S$ a code over $\mathcal{A}(A)$). $S$ is an unambiguous code if and only if $\overline{S}^{C}$ is an unambiguous code.*

□

The results in Proposition 2 and Corollary 1 do not hold if we drop the hypothesis of completeness, as it is shown in the following example.

**Example 17** Let $A = S \cup C$ where $S = \{ac, a, d\}$ and $C = \{\{a\}, \{c, d\}\}$. $S$ is unambiguous but $\overline{S}^{C} = \{A, AC, C\}$ is not unambiguous. In fact, $S$ is not complete.

Its completion is ${}^{C}S = \{ac, a, c, d\}$ which is not unambiguous.

□

Suppose now that we have an interpretational scheme $A = S \cup C$ and a text $\mathcal{T}$. We can represent $\mathcal{T}$ in $S$ or in $C$. Then we can apply $\overline{C}^{S}$ and $\overline{S}^{C}$ on these two representations. The following theorem states that the results obtained are identical modulo a canonical equivalence.

**Definition 14** Two sets of symbols $A$ and $A'$ are *equivalent*, $A \sim A'$ if there exists a bijective function $\phi : A \rightarrow A'$ called *equivalence*. If $\phi : A \rightarrow A'$ is an equivalence and $\mathcal{T}^{A} = a_1 ... a_n$ a text over $A$ we define $\phi(\mathcal{T}^{A}) = \phi(a_1)...\phi(a_n) = \mathcal{T}^{A'}$. Confusion is avoided by the superscripts denoting the alphabet the texts are constructed upon.

Two texts $\mathcal{T}, \mathcal{T}'$ over the alphabets $A$, $A'$ are *equivalent*, $\mathcal{T} \sim \mathcal{T}'$, if $A \sim A'$ with equivalence $\phi$ and $\phi(\mathcal{T}) = \mathcal{T}'$

□

**Theorem 1** *Let $A = S \cup C$ an unambiguous complete scheme. Then there exists a canonical equivalence between the representation of $\overline{\mathcal{T}}^{C}$ by $\overline{S}^{C}$ and the representation of $\overline{\mathcal{T}}^{S}$ by $\overline{C}^{S}$ for any text $\mathcal{T}$ over the alphabet of $A$.*

*Proof:* We need to show that here exists a canonical equivalence $\phi : \overline{S}^{C} \rightarrow \overline{C}^{S}$ such that

$$\phi\left( \overline{\overline{\mathcal{T}}^{C}}^{\overline{S}^{C}} \right) = \overline{\overline{\mathcal{T}}^{S}}^{\overline{C}^{S}}$$

for any text $\mathcal{T} \epsilon \psi(S)^{*}$

Define $\phi : \overline{S}^{C} \rightarrow \overline{C}^{S}$ as

$$\phi(s) = c, \psi(c) = \{t \epsilon S | \overline{\psi(t)}^{C} = \psi(s)\} \tag{2.3}$$

Let $\mathcal{T} \epsilon \psi(S)^{*}$ and $\overline{\mathcal{T}}^{S} = s_1 ... s_m$ is the unique encoding of $\mathcal{T}$ by $S$: $\mathcal{T} = \psi(s_1)...\psi(s_m)$ and so $\overline{\mathcal{T}}^{C} = \overline{\psi(s_1)}^{C} ... \overline{\psi(s_m)}^{C}$. By Definition 12 $(a)$ of $\overline{S}^{C}$ we have $\overline{\psi(s_i)}^{C} \in \psi(\overline{S}^{C}), 1 \leq i \leq m$. Hence

$$\text{there exists } \tilde{s}_i \in \overline{S}^{C} \text{ such that } \overline{\psi(s_i)}^{C} = \psi(\tilde{s}_i), 1 \leq i \leq m \tag{2.4}$$

So $\tilde{s}_1...\tilde{s}_m$ is an encoding of $\overline{T}^C$ by $\overline{S}^C$. By Proposition 2 it is the only one, so $\tilde{s}_1...\tilde{s}_m = \overline{\overline{T}^C}^{\overline{S}^C}$.

On the other hand, by Definition 12 ($b$),

$$\text{there exists } \tilde{c}_i \in \overline{C}^S \text{ such that } s_i \epsilon \psi(\tilde{c}_i),\ 1 \leq i \leq m \tag{2.5}$$

hence if we consider again $\overline{T}^S = s_1...s_m$ we have $\tilde{c}_1...\tilde{c}_m = \overline{\overline{T}^S}^{\overline{C}^S}$.

We only need to show that $\phi(\tilde{s}_i) = \tilde{c}_i, 1 \leq i \leq m$. By (2.3) we have that

$$\phi(\tilde{s}_i) = c,\ \psi(c) = \{t \epsilon S | \overline{\psi(t)}^C = \psi(\tilde{s}_i)\}$$

(2.4) implies that $\psi(\tilde{s}_i) = \overline{\psi(s_i)}^C$ hence $s_i \epsilon c$. Since $\overline{C}^S$ is an equivalence relation, (2.5) implies that $c = c_i$

□

### 2.2.4 The Alphabet

The canonical equivalence in the previous theorem permits to define the action of an unambiguous complete scheme on a text. In fact we can conventionally change into identical symbols the ones in the $\overline{X}$ of $\overline{S}^C$ and $\overline{C}^S$ which are in correspondence through the equivalence $\phi$ constructed above. In that situation Theorem 1 guarantees that the result of applying $\overline{S}^C$ on $\overline{T}^S$ is the same as that of applying $\overline{C}^S$ on $\overline{T}^C$. This leads us to improve Definition 4: when we live in a semantic world all the symbols have meaning. The following definition is also an attempt to formalize the notion of *alphabet*, intended in the linguistic sense of the term.

**Definition 15** An *alphabet* is a triple $(\overline{X}, \psi, A)$ where $A = S \cup C$ is a complete unambiguous scheme, $\overline{X}$ is a set of symbols, and $\psi$ a bijective function $\psi : \overline{X} \hookrightarrow \overline{S}^C$ (or equivalently $\psi : \overline{X} \hookrightarrow \overline{C}^S$).

□

We will abuse the notation and conventionally write $A = S \cup C$ even for alphabets It is understood that the set of symbols in $A$ is the one of $\overline{S}^C$ $(\overline{C}^S)$. It is also understood that alphabets are unambiguous and complete.

**Definition 16** Let $A = S \cup C$ be an alphabet. An *interpretation* $\overline{\mathcal{T}}^A$ of a text $\mathcal{T}$ under the alphabet $A$ is the text over $A$ obtained by applying $\overline{S}^C$ $(\overline{C}^S)$ on $\overline{\mathcal{T}}^S$ $(\overline{\mathcal{T}}^C)$. *Explanation* is the inverse operation of interpretation. Explanation can be multivalued.
□

**Example 18** Let $\mathcal{T}$ be a text formed by juxtaposition of sentences on $\mathcal{W}$ generated by our natural language grammar, take, for instance,

$$\mathcal{T} = T\ ADJ\ N\ TV\ ADJ\ N\ P\ ADJ\ N\ T\ N\ TV\ N\ P\ N\ TV\ to\ V\ D$$

Change the grammar by modifying the <PN> and the <TOVO> production in the following way:

<PN> ::= *P* <NSTG1>
<TOVO> ::= *to* <LVR> <OBJ1>

and by adding the productions

<NSTG1> ::= <LNR>
<OBJ1> ::= <NSTG> | <TOVO> | *null*

Construct the semantic universe, $\mathcal{Z}$, $\mathcal{S}$, and $\mathcal{T}$ as in Example 6. In the following two example we will abuse the overline superscript notation introduced. Thanks to Theorem 1 it is possible to collapse a large stack of overline superscripts, e.g.

$$\overline{\overline{\overline{\mathcal{T}}^A}^B}^C = \overline{\mathcal{T}}^C$$

1. $\mathcal{T}$ interpreted in $\mathcal{S}$ is $\overline{\mathcal{T}}^{\mathcal{S}}$ = S S S

2. $\mathcal{T}$ interpreted in $\mathcal{Z}$ is $\overline{\mathcal{T}}^{\mathcal{Z}}$ = SUBJ VERB OBJ SUBJ VERB SUBJ VERB OBJ

□

**Example 19** The entries of a dictionary of a language is an alphabet on the context of all the roots and suffixes. The words are classified by chosing one representative for all the declensions of a word. The dictionary itself describes the semantics of the entry by explaining some of its syntaxes in the context of all the words of a language.
□

### 2.2.5 Hierarchies of Alphabets

Let us now consider our formulation of the learning problem as the construction of a semantics for the reality under observation. We are now able to understand how to take this definition: the goal is to construct a semantic universe which contains the symbols in the text observed. We are now in the situation where there are only two constructors -remember that in the next chapter we will appreciate their strength- so it is easy to devise a way to proceed.

- Construct, *in an appropriate way*, an alphabet over the set of symbols present in the text, i.e., construct a classification and a code and, for each set or string $s$ constructed, create a symbol $\overline{s}$ and assign $\psi(\overline{s}) = s$;
- Interpret the text under the alphabet constructed and proceed both recursively and at the same level by constructing another *appropriate* alphabet.

The goal of the next chapter is to determine what *appropriate* should be taken to mean. This section clarifies what are good families of alphabets.

Since the reference is the inverse function of the semantics, if one constructs a well behaved reference on a universe of symbols $\mathcal{U}$, that will define a semantics on $\mathcal{L}^+(\mathcal{U})$.

A *good* family of alphabets is one that defines a well behaved reference, i.e. a family

$$\{A_i = (\overline{A}_i, \psi_i, \mathcal{A}(A_i))\}_{i \epsilon I}$$

such that for every $i, j \epsilon I, i \neq j$, then $s \epsilon \overline{A_i} \cap \overline{A_j}$ implies $\psi_i(s) = \psi_j(s)$. In fact, in this situation we can construct a reference: let $\Sigma = \bigcup_{i \epsilon I} \overline{A}_i$ and $\mathcal{U} = \bigcup_{i \epsilon I} \mathcal{A}(A_i)$; define $\psi : \Sigma \hookrightarrow \mathcal{L}(\mathcal{U})$, in such a way that $\psi(s) = \psi_i(s)$ if $s \epsilon \overline{A}_i$. Then there exists an extension of $\mathcal{U}$ and an extension $\phi$ of $\psi^{-1}$ such that $(\mathcal{U}, \phi, \Omega)$ is a semantic universe.

In Chapter 3, however, we focus mainly on hierarchical families of alphabets:

**Definition 17** A family $\{A_i\}^n_{i=0}$ of alphabets is *hierarchical* if it is a semantic universe and $\mathcal{A}(A_i) \subseteq A_{i-1}$, $1 \leq i \leq n$.

□

Hierarchical families have the nice property pointed out in the following

**Proposition 3** *Let* $\{A_i\}^n_{i=0}$ *be a hierarchical family of alphabets. Then* $s \epsilon \overline{A_i} \cap \overline{A_j}$ *implies* $\psi_i(s) = \psi_j(s) = s$, *for every* $0 \leq i \neq j \leq n$.

□

## 2.3 Structures

Structure is one of the universal notions common to several scientific disciplines. This section is a small digression to relate the structures that we have just introduced with algebraic and computational structures. We will not do the same with structures in logic. This field is so closely related that even a quick glimpse to it, is far too laborious a work. As far as algebraic structures are concerned, we will be content to note that monoids, sets with an associative operation and a unit, are probably the most studied structures in algebra: other structures are constructed in terms of those. The interesting relation with our construction is that any monoid can be represented as a free monoid (the universal language in computer science jargon) and an equivalence relation on its words [47].

### 2.3.1 Computational Structures

In Computer Science the structure is represented by the program, or by any other equivalent computational unit like grammar, Turing Machine and recursive function. The question that one might ask is whether alphabets and schemes are equally expressive.

The answer comes with Theorem 2: in the situation where $\Omega$ only contains the operation of string formation and set formation, the interpretational scheme can play the same role as the program in an universal computational scheme.

The computational protocol that we describe uses schemes to perform the operations (OPS) of generalization (GEN), encoding (ENC), specialization (SPC), and spelling (SPL). However we need to modify their definition a little bit and allow OPS to perform also on only one symbol, or one substring of the text. Other computational protocols can be invented using alphabets and their actions in a different way. The one we propose is meant to simulate exactly the behavior of a general phrase structure grammar (of type 0). It is still an open problem whether Theorem 2 is valid when in the case when OPS are applied exactly the way they were introduced in Definitions 9 and 10, but we shouldn't be too worried and miss the forest for the tree.

Our simulating device uses two schemes $A$ and $B$. $A$ is used indifferently and nondeterministically to perform (GEN), (ENC), (SPC), and (SPL) on a text $\mathcal{T}$. $B$ is used to perform only (SPC) and (SPL). We select a subset $T \subseteq \mathcal{A}(A) \cup \mathcal{A}(B)$ of terminal symbols. A computation is the performance of the operations (OPS) -with the restrictions above- on a given text $\mathcal{T}$. The computation is in a final state when the text $\mathcal{T}$ contains only terminal symbols.

**Lemma 1** *Let $G$ be a general phrase structure grammar. Let $G'$ the grammar obtained from $G$ by deleting all the productions of the kind $s \rightarrow t$ where $s$ and $t$ are single symbols (terminal or variable) and by substituting $t$ for $s$ in each production of $G$. Then $L(G) = L(G')$.* (See Theorem 4.4 [46])

□

**Theorem 2** *Given any general phrase structure grammar $G$ there exists a computational setting as described above which simulates the derivation by $G$ of words in the language $L(G)$.*

*Proof :* By Lemma 1 we can suppose that $G$ contains no productions of the form $s \rightarrow t$ where $s$ and $t$ are distinct singleton symbols. Let all the productions in $G$ be

$$w_i \longrightarrow v_{i,j} \tag{2.6}$$

where $w_i$ and $v_{i,j}$ are words on the set of symbols of $G$,$1 \leq j \leq k_i$, $1 \leq i \leq n$.

Let $A = \{a_1, ..., a_n, b_1, ..., b_n\}$ where $\psi(a_i) = w_i$; $\psi(b_i) = \{a_i, c_i\}$ and $B = \{c_i, ..., c_n\}$ where $\psi(c_i) = \{d_{i,1}, ..., d_{i,k_i}\}$ and $\psi(d_j) = v_{i,j}$. All the new symbols introduced $a_i, b_i, c_i, d_i$ are distinct form the symbols of $G$. Moreover, let the terminal symbols of the device be the same as the terminal symbols of $G$.

It is easy to see that for each use of a production rule of the $G$ then there is a sequence of OPS in the order [ENC GEN SPC SPL] which achieves the same result. Moreover if the device starts on the text composed of the only starting symbol of the grammar, it will only generate words of $L(G)$.

□

The following example shows that an alphabet used as a computational device can sometime provide a more compact definition of a language.

**Example 20** Consider the context sensitive language $x^n z^n y^n, n \geq 0$. It can be generated by the grammar:

$$S \rightarrow xS\lambda\mu \quad S \rightarrow xz\mu \quad S \rightarrow []$$
$$\mu\lambda \rightarrow \lambda\mu \quad z\lambda \rightarrow zz \quad \mu \rightarrow y$$

It is just a matter of a little bit of work to see that we need $O(n^2)$ steps to get $x^n z^n y^n$.

Let us go back to our usual mode of applications of OPS as in Definitions 9 and 10. The following alphabet

$$S \xrightarrow{\psi} 1x1z1y \quad 1x \xrightarrow{\psi} 1xx$$
$$1y \xrightarrow{\psi} 1yy \quad 1z \xrightarrow{\psi} 1zz$$

obtains $1x^n1z^n1y^n$ in $O(n)$ applications of SPL. One application of SPL with the following alphabet will yield $x^nz^ny^n$.

$$1x \xrightarrow{\psi} x$$
$$1y \xrightarrow{\psi} y \quad 1z \xrightarrow{\psi} z$$

Note that even if we counted the spelling of each symbol as one operation, still we would need only $O(n)$ of them. Of course the same could be achieved with grammars if we had rules organized in sets, with the requirement that if one rule is applied then all the other in the same set must be applied.
□

## 2.4 Open Problems and Future Work

### 2.4.1 Multidimensional Codes

We might want to explore if it is possible to extend this ideas in order to apply them also the the analysis and learning of the visual image. As a first step we will consider only images that have a discrete structure but are not simply texts, i. e., totally ordered sets of symbols. Many mathematical texts can be taken as examples of such images. For example:

$$\int_{\Sigma} \frac{(a_1^{\beta_1} + a_2^{\beta_2})}{A_{22}} d\sigma$$

For some reason its expression in Latex©does not look as expressive due to its nested parentesis and functional symbols:

```
\[ \int_{\Sigma} \frac{(a_{1}^{\beta_{1}}+ a_{2}^{\beta_{2}})}{A_{22}}
d\sigma \]
```

The following definitions are the first step of an attempt in the direction of treating those texts without having to impose a total order on its symbols.

**Definition 18** A *texture* is a metric space $(\Upsilon, d)$ where $d$ is a metric. The elements of $\Upsilon$ are called *positions*.

□

It follows immediately that a texture is a topological space, with the topology induced by the metric $d$. The topology is defined in the usual way by considering the balls centered in every point as a base for the neighbourhoods in the usual way,.

Hierarchical textures play an important role in the development our approach. Their definition is based on the notion of *ultrametric*, which is simply a metric where the triangular inequality is valid in the stronger form $d(x,z) \leq max(d(x,y), d(y,z))$

**Definition 19** A texture $(\Upsilon, d)$ is *hierarchical* if there exists an ultrametric $d'$ such that the topology induced by $d'$ on $\Upsilon$ is the same as the topology induced on $\Upsilon$ by $d$.

□

**Definition 20** A *hypertext* over an alphabet $A$ is a couple $(\Upsilon, \phi)$ where $\Upsilon$ is a connected (in topological terms) texture and $\phi$ a function $\phi : \Upsilon \rightarrow A$.

□

The definition of hypertext allows a generalization of all the concepts in the the previous sections. In fact, we can substitute hypertexts for texts and strings in the previous sections: all the concept we defined would still make sense and all the facts we proved would still be true. Just a little bit of attention is needed to take care of the metric in the texture of an hypertext when we paste together different pieces.

# Chapter 3

# Measurement

Most of our definitions of learning have a precise meaning at this point. We now assume that a learning observer $A$ is an interpretational scheme $(X, X)$ where $X$ is all the symbols that $A$ has previously constructed. $A$ is an interpretational scheme, or equivalently an alphabet, simply means that $A$ will interpret, or encode, $\mathcal{T}$ in terms of the symbols that $A$ "knows". Let us do that and consider $\overline{\mathcal{T}}^A$. Whatever was unknown in $\mathcal{T}$ is now lost (see Remark 4) and what is left are only symbols in $A$. So we can consider $\mathcal{T}$ as a text over $X$ or, equivalently, $A$.

The goal of the present chapter is to supply our learning observer with a means of measurement for its representation of the reality under observation. The object to be represented is a text $\mathcal{T}$. We will assume almost no requirement for the measurement function, which will be called a *measure*. The nature of the measure, whether it depends on internal parameters or it is given from an external agent, will determine if learning is supervised or unsupervised.

In the following chapter we will use the general notion of measure introduced to construct better representations of the text. It is obvious that one can use more than one measure concurrently.

## 3.1 Efficiency of Encoding

A possible measure can be construed form the following example. Assume that $A$ has an internal representation for a portion of $\mathcal{T}$ in a given bounded memory space. The first problem that arises is that of finding a smart way to encode the text $\mathcal{T}$ in order to have an internal representation of a larger portion. One way to do that is to construct another interpretational scheme $(B, A)$, and to represent $\overline{\mathcal{T}}^B$. In this situation the desired measure is the size of the encoding. As we will see in the next section we can quantify the gain obtained in this way.

First, in order to understand what a good encoding is, we need to introduce some notational conventions and to borrow a theorem from Information Theory.

### 3.1.1 Definitions and Notational Conventions

Unless otherwise stated $\mathcal{T}$ will be a text [1] over an alphabet $A = \{a_i\}_{i=1}^n$.

1. A *measure* or *cost* $|.|$ is a function from the set $B$ of all the texts and classes over an alphabet to the real numbers, such that, for each $a, b \epsilon B$,

   $a)$ $\psi(a) \subseteq \psi(b) \Rightarrow |a| \leq |b|$

   $b)$ $|a \cup b| \leq |a| + |b|$

   $\psi$ is as in Definition 4. The relation $\subseteq$ in $a)$ is overloaded to mean "substring" or "subset", whatever applies. No relation is assumed between a string and a set. Analogously in $b)$ $\cup$ is overloaded and denotes string concatenation or set union.

   An example of measure is the length for texts and the cardinality for classes.

2. $\mathcal{T}_N$ is the prefix subtext of $\mathcal{T}$ such that $|\mathcal{T}_N| = N$. If $N \geq |\mathcal{T}|$ we define $\mathcal{T}_N = \mathcal{T}$.

[1] The default assumption is that texts are finite. Most of the results to follow will be applicable also to infinite texts given appropriate care and the use of limit operations.

3. $\#^a(\mathcal{T})$ is the text obtained by deleting all symbols but the symbol $a$ from $\mathcal{T}$. Note that if the measure is length, then $|\#^a(\mathcal{T})|$ is the number of occurrences of the symbol $a$ in $\mathcal{T}$.

4. A probability $p^{\mathcal{T}}$ is defined on $A$ through $\mathcal{T}$ by means of the formula

$$p^{\mathcal{T}}(a) = \lim_{N\to\infty} \frac{|\#^a(\mathcal{T}_N)|}{|\mathcal{T}_N|}$$

   For finite texts the expression above does not cause any problem. We assume that infinite texts are nice and this limit always exists.

   In general a probability is just a function that is defined in term of the cost. However, if the measure is length then the induced probability measures the relative frequencies in $\mathcal{T}$ of the symbols in $A$.

5. If $C$ is a classification over $A$, we can also consider the conditional probability

$$p^{\mathcal{T}}(a|c) = \begin{cases} 0 & \text{if } a \notin c \\ \frac{p^{\mathcal{T}}(a)}{p^{\overline{\mathcal{T}}^C}(c)} & \text{if } a \in c \end{cases}$$

6. An *n-ary code* is a code over an alphabet with $n$ symbols.

7. Let $S$ be an unambiguous code over $A$ and $\overline{\mathcal{T}}^S$ a text over $S$. $S$ is *optimal* for $\overline{\mathcal{T}}^S$ if the *expected cost* of one symbol $s\epsilon S$,

$$\langle S\rangle^{\overline{\mathcal{T}}^S} = \sum_{s\epsilon S} p^{\overline{\mathcal{T}}^S}(s)\,|\psi(s)|$$

   is minimal. Note that properties *a*) and *b*) in 1. imply that when $S$ is optimal for a text $\mathcal{T}'$ the spelling $\underline{\mathcal{T}}'_S$ has minimal cost.

8. Let $C$ be a classification on $A$. The *expected cost* of $C$ is

$$\langle C\rangle^{\overline{\mathcal{T}}^C} = \sum_{c\epsilon C} p^{\overline{\mathcal{T}}^C}(c)\,|\psi(c)|$$

9. $OPT^n(\mathcal{T})$ is the cost of $\underline{\mathcal{T}}_S$ where $S$ is an optimal code over an alphabet with $n$ symbols.

10. The *compression coefficient* $\lambda_{\mathcal{T}}(A)$ of an alphabet $A$ for a text $\mathcal{T}$ is

$$\lambda_{\mathcal{T}}(A) = \lim_{N \to \infty} \frac{\left| OPT^n(\overline{\mathcal{T}}_N^A) \right|}{|OPT^n(\mathcal{T}_N)|}$$

11. The $n$-ary *information*, or *entropy* of a text $\mathcal{T}$ over an alphabet $A$ is the expression

$$i(\mathcal{T}) = -k \sum_{a \epsilon A} p^{\mathcal{T}}(a) \log p^{\mathcal{T}}(a)$$

where $k = 1/\log n$. All the logarithms are in base $e$.

12. The $n$-ary *information*, or *entropy*, of a class c of elements in $A$ is

$$i(c) = -k \sum_{a \epsilon A} p^{\mathcal{T}}(a|c) \log p^{\mathcal{T}}(a|c)$$

**Theorem 3** *The expected cost of a symbol in $\mathcal{T}$ under an optimal* n*-ary code is equal to the* n*-ary information of $\mathcal{T}$.*

□

The previous fundamental theorem of information theory shows that the expression for *information* is actually a quantification of the same commonly used concept. However, it is often more intuitively rewarding if it is taken to quantify the information that *can* be contained in a text.

We have formulated this theorem in its generalized form, where the cost function does not have to be the length. The same proof as in [49], is still valid, *mutatis mutandis*, in this more general situation.

### 3.1.2 Block Coding

One way to optimize the encoding process is *block coding* a well known technique of Information Theory. The idea is very simple: instead of encoding each symbol by itself, one encodes blocks of symbols. In other words, before encoding, we choose a code $S$ over $X$ and consider $\overline{\mathcal{T}}^S$, the interpretation of $\mathcal{T}$ by $S$. Let us denote

with $\langle S \rangle^{\overline{\mathcal{T}}^S}$ the expected cost of the encoding of one symbol of $S$ in $\mathcal{T}$. Then the expected cost of $\overline{\mathcal{T}}_N^S$ is

$$N' = \frac{N}{\langle S \rangle^{\overline{\mathcal{T}}^S}} \tag{3.1}$$

By Theorem 3 we know that the expected cost of an optimal encoding of $\overline{\mathcal{T}}^S$ is

$$OPT^n(\overline{\mathcal{T}_N}^S) = N' \cdot i(\overline{\mathcal{T}}^S) \tag{3.2}$$

(Let $n$ be the number of symbols of the alphabet of $T$).

Putting 3.1 and 3.2 together we have

$$OPT^n(\overline{\mathcal{T}_N}^S) = N \frac{i(\overline{\mathcal{T}}^S)}{\langle S \rangle^{\overline{\mathcal{T}}^S}} \tag{3.3}$$

Since $OPT^n(\mathcal{T}_N) = N$ then we have proved the following

**Proposition 4** *Let $S$ and $\mathcal{T}$ be a code and a text over an alphabet $A$ respectively. The compression coefficient of $S$ for $\mathcal{T}$ is*

$$\lambda_{\mathcal{T}}(S) = \frac{i(\overline{\mathcal{T}}^S)}{\langle S \rangle^{\overline{\mathcal{T}}^S}}$$

□

**Example 21** Let us consider the Morse code $\mathcal{M}$ as in Example 13. Let $\mathcal{T}$ be English encoded with the Morse code (in our terminology we should say *spelled* with the Morse code). Then

| | - | · | *space* |
|---|---|---|---|
| $p$ | 0.2875 | 0.4297 | 0.2868 |

We have $i(\mathcal{T}) = 1.0785$; $i(\overline{\mathcal{T}}^{\mathcal{M}}) = 2.8902$; $\langle \mathcal{M} \rangle^{\overline{\mathcal{T}}^{\mathcal{M}}} = 3.5361$. Finally

$$\lambda_{\mathcal{T}}(\mathcal{M}) = 0.8173$$

□

As we have seen in the introduction the expression *there is structure* in a text, means that the text can be compressed. The following example is meant to show that a text is structured whenever there is a string of symbols, of any length, which never appears in the text.

**Example 22** Let $\mathcal{T}$ be a text over an alphabet $A$, $|A| = n$, and $S$ an unambiguous code over $A$, $|S| = m$. Let us suppose that no information is available about the probabilities $p^{\mathcal{T}}(A)$ and $p^{\mathcal{T}}(S)$ but that $\langle S \rangle^{\overline{\mathcal{T}}^S} = l$. Since we have no information about the probability, we take it to be the uniform distribution. In this situation $\lambda_{\mathcal{T}}(S)$ is at most $\frac{\log m}{l}$ since the information is maximal when $p^{\mathcal{T}}$ is the uniform distribution [49]. By the same token, $\lambda_{\mathcal{T}}(A) = \log n$. So whenever $n^l \geq m$ it is convenient to use $S$. If all the strings of $S$ have the same length $l$, then the text is compressible when not all the strings of length $l$ are represented in the text.
□

**Example 23** *The Discovery of Phrase Structure*

The coefficient $\lambda$ can be employed in many different ways to the study of language. In this section we will use it to evaluate the performance of algorithm N in [10] given in Figure 3.1.

We considered a text consisting of 100 words randomly generated using the natural language grammar given in Section 1.4.4. If we run twice the algorithm N in [10] on such a set of sentences (which correspond to finding two hierarchical levels of codes) we get the code

$$S = \left\{ \begin{array}{cccc} (to\ V) & (P\ T\ N) & (T\ ADJ\ N) & (P\ N) \\ (P\ ADJ\ N) & (to\ D\ V) & (ADJ\ N) & (D) \\ (N) & (T\ N) & (P\ T\ ADJ\ N) & (TV) \end{array} \right\}$$

In the following table, we can compare the $\lambda$ value of the alphabet $A$ of terminal symbols, $S$, and $S_2$ consisting of all the digrams appearing in the text and all the word-ending monograms (to guarantee parsing of sentences with an odd number of components).

| | $A$ | $S$ | $S_2$ |
|---|---|---|---|
| $\lambda$ | 2.2701 | 1.676 | 2.183 |

**Algorithm N**

$N = \emptyset, T = \emptyset$

Forall sentences $w$

Consider the terminal symbol of $w$, say $t$;

a) if $t \in T$ then factorize[2] $w$ in strings ending (and not containing) elements of $T$, call the set of these strings $V$ and set $N = N \cup \{V\}$.

b) if $t \notin T$ then factorize the elements of $N$ in strings ending with and not containing $t$; further, set $T = T \cup \{t\}$, factorize $w$ in strings ending with (and not containing) elements of $T$ and add them to $N$.

end forall

Figure 3.1: Algorithm N

As we see there is a sensible difference in the gain obtained using Algorithm N. Moreover, if we notice that the grammar chosen is one which describes (even if incompletely) the word classes phrase structure of English, we find that the code-words discovered by Algorithm N coincide with the ones which a person (not only a linguist) would normally consider to be the elementary strings [42] of the phrase structure.

This results sustains the conjecture that guided our work on the subject: $\lambda$ reaches a local minimum over codes which could be and have been discovered by means of other "reasonable" considerations. The process which guides language use is the same at different levels and linguistic ability is the superimposition of the same operations on different substrata.

□

[12] For example, *abcaadbcd* factorized by $T = \{c, d\}$ yields $V = \{abc, aad, bc, d\}$

### 3.1.3 Class Coding

While there is no compression gain for a text $\mathcal{T}$ over an alphabet $A$ when using only a classification $C$ over $A$, there can be a gain when the classification is associated to a code. Let $B = S \cup C$ an alphabet over $A$. Let $\mathcal{T}' = \overline{\mathcal{T}}^C$ and $S' = \overline{S}^C$. It is easy to see that $\langle S \rangle^{\overline{\mathcal{T}}^S} = \langle S' \rangle^{\overline{\mathcal{T}'}^{S'}}$, so the expected cost of $\overline{\mathcal{T}_N}^B$ is as in 3.3.

Once again, by Theorem 3 the lower bound for the expected cost of an optimal encoding of one symbol of $\overline{\mathcal{T}_N}^B$ is

$$N' \cdot i(\overline{\mathcal{T}}^B) = N' \cdot i(\overline{\mathcal{T}'}^{S'}) \tag{3.4}$$

Note that in general $i(\overline{\mathcal{T}}^B) \leq i(\overline{\mathcal{T}}^S)$. However some information is lost. $i(\overline{\mathcal{T}}^B)$ is not sufficient to reconstruct $\mathcal{T}$ given $\overline{\mathcal{T}}^B$ because we are missing the information regarding what element of $\overline{C}^S$ is to be considered. Remember, in fact, that there can be more than one specialization under a classification. We will now quantify the extra information needed.

Theorem 1 shows that there is an equivalence between $\overline{C}^S$ and $\overline{S}^C$, so we can indifferently consider any of the two. Let $\overline{s} \epsilon \overline{S}^C, \psi(\overline{s}) = c_1 ... c_h$. The additional information to determine which symbol $c_i$ stands for is given by $i(c_i)$. Its expected value is

$$\langle C \rangle^{\overline{\mathcal{T}}^C} = \sum_{c \epsilon C} p^{\overline{\mathcal{T}}^C} i(c) \tag{3.5}$$

Then we have to multiply that value for $\langle S \rangle^{\overline{\mathcal{T}}^S}$, the expected cost of the codeword $\psi(\overline{s})$. We finally get that the information per symbol of $\overline{\mathcal{T}}^B$ is

$$i(\overline{\mathcal{T}}^B) + \langle S \rangle^{\overline{\mathcal{T}}^S} \langle C \rangle^{\overline{\mathcal{T}}^C} \tag{3.6}$$

We only need to multiply it by $N'$ to get the expected cost of an optimal encoding of $\overline{\mathcal{T}}^B$.

$$OPT^n(\overline{\mathcal{T}}^B) = N' \left( i(\overline{\mathcal{T}}^B) + \langle S \rangle^{\overline{\mathcal{T}}^S} \langle C \rangle^{\overline{\mathcal{T}}^C} \right) \tag{3.7}$$

Now, using 3.1 we get

$$OPT^n(\overline{\mathcal{T}}^B) = N\frac{i(\overline{\mathcal{T}}^B) + \langle S\rangle^{\overline{\mathcal{T}}^S}\langle C\rangle^{\overline{\mathcal{T}}^C}}{\langle S\rangle^{\overline{\mathcal{T}}^S}} \tag{3.8}$$

We have proved the following:

**Theorem 4** *Let $B = S \cup C$ and $\mathcal{T}$ be an alphabet and a text over an alphabet $A$. The compression coefficient of $B$ for $\mathcal{T}$ is*

$$\lambda_{\mathcal{T}}(B) = \frac{i(\overline{\mathcal{T}}^B)}{\langle S\rangle^{\overline{\mathcal{T}}^S}} + \langle C\rangle^{\overline{\mathcal{T}}^C}$$

□

The reader can get an intuition about how the classification allows a further gain in compression by considering that it causes a decrease in the number of symbols of the alphabet that is being constructed. When the number of symbols decreases, the entropy function "usually" decreases. Note that Proposition 4 is a particular case of Theorem 4 when the classification is the trivial one where each class contains only one symbol.

**Example 24** Let us consider the Morse code as in Example 21 with a slight variation. Let us suppose the that the source and the receptor of the Morse signal are not well trigged so that the receptor sees the following signals: $-$ $+$ $\cdot$ $\circ$ *space*, even if the source is transmitting only $-$ , $\cdot$, *space*. When the source transmits a $-$ the receptors gets either a $-$ or a $+$ with same probability. Analogously, when the source transmits a $\cdot$ the receptors gets either a $\cdot$ or a $\circ$ with same probability. The *space* is received correctly.

Let us consider an English text $\mathcal{T}$ and let $T$ be a possible way $\mathcal{T}$ looks to the receptor. $T$ is a text over the alphabet M= $\{-, +, \cdot, \circ, space\}$. If the English text $\mathcal{T}$ is long enough, we have

| | - | + | · | ○ | *space* |
|---|---|---|---|---|---|
| $p$ | 0.1431 | 0.1431 | 0.2140 | 0.2140 | 0.2857 |

We have

$$i(T) = \lambda_T(\mathbf{M}) = 1.5743 \tag{3.9}$$

It is obvious that if $C$ is the classification $\{\{-,+\},\{\cdot,\circ\},\{space\}\}$ then $\overline{T}^{C}$ is the same as the spelling $\underline{\mathcal{T}}_{\mathcal{M}}$. Consider the completion of $\mathcal{M}$ by $C$ and call it $M'$, $M' = {}^{C}\mathcal{M}$. There are 276 elements in $M'$.

$i(\overline{T}^{M'}) = 4.6481$ and the compression coefficient $\lambda_T(M') = 1.3247$. Note that it is convenient to use $M'$ because $\lambda_T(M') < \lambda(\mathbf{M})$ in expression 3.9.

Now, the alphabet $E = C \cup M'$ acts on $T$ as the english alphabet, so $\overline{T}^{E} = \mathcal{T}$, ($\mathcal{T}$ is the english text we started with). We can compute $\langle C \rangle^{\overline{T}^{C}}$ with the formula 3.5. We get

$$\langle C \rangle^{\overline{T}^{C}} = 0.4944$$

. On the other end from Example 21 we recall

$$\frac{i(\overline{T}^{E})}{\langle M' \rangle^{\overline{T}^{M'}}} = 0.8174$$

(note that $\langle M' \rangle^{\overline{T}^{M'}} = \langle \mathcal{M} \rangle^{\overline{T}^{\mathcal{M}}}$).

It follows from Theorem 4 that $\lambda_T(E) = 1.3118$. The use of the classification $C$ allows a further gain over the use of the code $M'$.

## 3.2 The Objective Viewpoint

Until now we have embraced a *subjective* viewpoint: no assumption were made on the text under observation. In this section we will take a different standpoint and assume that the text is meant for communication ends, as in the case where $\mathcal{T}$ is an utterance in a natural language. In this new situation it is assumed that the text is constructed in such a way to exhibit properties of optimality and that our learning system attempts to adjust to that external optimal shape.

In a less general setting, we will be able to see that constructing an interpretational scheme can be viewed as an evolution toward equilibrium of a physical

system. In this way we will find an intuition that the expressions that we have found are related in many ways to the expressions of physical concepts as *energy*, *entropy*, *temperature*.

### 3.2.1 The Boltzman-Gibbs Distribution

The interpretational scheme $B$ that we will consider now is simply a code. We choose to do this because the general case, when $B = S \cup C$ and $C$ is a nontrivial classification, requires more care and will result in a less fluent exposition.

Let us consider the set $\mathcal{S}$ of all the unambiguous codes over an alphabet $A$. A text $\mathcal{T}$ over $A$ can be thought of as a partial function

$$\begin{array}{ccccc} \mathcal{T} & : & \mathcal{S} & \longrightarrow & \mathcal{P} \\ & & S = \{s_k\} & \longrightarrow & q = \{q_k\} \end{array}$$

where $q_k$ is the frequency of the symbol $s_k$ in the text. $\mathcal{T}(S)$ can be undefined if $S$ does not parse $\mathcal{T}$. See also Remark 4. It is easy to observe that the condition of unique decipherability of $S$ implies that $\mathcal{T}$ is injective. It should be noted that this condition could be weakened by making assumptions on the parsing procedure for the codewords, instead of the nature of the code. If the conventions for parsing are "good", for instance it favors "the longer the better" principle, then $\mathcal{T}$ will still be injective.

In the previous section we have found a way to evaluate the choice of the code $S \epsilon \mathcal{S}$. In the present situation we can reason in a different way, since we have assumed that the text $\mathcal{T}$ is meant for communication purposes.

We invoke the formula

$$|s_k| = -\frac{\log p_k}{\beta} \tag{3.10}$$

where $\beta = \log n$. This formula relates the cost $|s_k|$ of transmitting signal $s_k$ in some optimal $n$-ary code and its probability of occurrence $p_k$. For the sake of our reasoning we need to mention that besides its information theoretical deriva-

tion based on the optimality condition (see Theorem 3)[3], the same formula has been mathematically established by [61] in other ways following diachronic and synchronic considerations on word formation.

For all symbols $s_k$ let $\psi(s_k) = s_1^k ... s_{i_k}^k$ We will relate cost $l_k$ of the string $s_k$ computed with the formula

$$l_k = \sum_{j=1}^{i_k} \left| s_i^k \right|$$

to its assigned cost $|s_k|$ by means of the relation

$$|s_k| = l_k + L \tag{3.11}$$

$L$ is needed as a normalizing factor as we will see in a moment.

If cost is simply length, as was suggested in [61], the expression above is justified by the consideration that cost is best interpreted as the time $l_k$ necessary to "read" (in a generalized sense) the string plus the time $L$ for recognizing its end which in our case, as opposed to the one in [61], is not marked by any special symbol.

From 3.10, 3.11, and the condition $\sum p_k = 1$ we get

$$p_k = \frac{e^{-\beta l_k}}{Z} \tag{3.12}$$

where $Z = e^{\beta L} = \sum e^{-\beta l_k}$. 3.12 is the Boltzmann-Gibbs distribution where the length of a codeword stands for its energy. The derivation above should convince us that we can expect the codewords to be distributed according to the Boltzmann-Gibbs distribution if we consider that once they are established they can be used as an alternative alphabet (as an example of real language consider Chinese) and that the choice in their use is not given to the language user.

We can now define the map

$$\begin{array}{ccccc} \mathcal{G} & : & \mathcal{S} & \longrightarrow & \mathcal{P} \\ & & S = \{s_k\} & \longrightarrow & p = \{p_k = \frac{e^{-\beta l_k}}{Z}\} \end{array}$$

[3]This formula is *not* a consequence of Theorem 3 but it is consistent with it in a very particular way: is constitutes the "best" (as far as basic information theory is concerned) interpretation of Theorem 3

and we can use Kullback *cross-entropy* [53]

$$K(q,p) = \sum q \log \frac{q}{p}$$

which measures the "information for discrimination" between the distributions $q$ and $p$, to define a measure of how close a code $S$ is to an optimal code.

Our learning mechanism will minimize this distance by minimizing the function $K(\mathcal{T}(S), \mathcal{G}(S))$.

The intuitive explanation relies on the fact that Kullback cross-entropy can be thought of as directed distance (in fact, besides as *relative entropy* it is also known as *directed divergence*) from the distribution $q$ to $p$. We should mention that the principle of cross-entropy minimization is a generalization of Jaynes principle of maximum entropy [48] which can be applied when a prior distribution $p$ that estimates the distribution $q$ is given in addition to the constraints. A detailed study and a bibliography on its applications which range from statistics to pattern recognition and spectral analysis can be found in [79,80].

**Example 25** Let us consider again the Morse code $M$ as in Example 21. If $\mathcal{T}$ is any long enough english text the vector $\mathcal{T}(\mathcal{M})$ is

| | a | b | c | d | e | f | g | h |
|---|---|---|---|---|---|---|---|---|
| $p$ | 0.0778 | 0.0128 | 0.0298 | 0.0417 | 0.1070 | 0.0251 | 0.0179 | 0.0576 |
| | i | j | k | l | m | n | o | p |
| $p$ | 0.0750 | 0.0059 | 0.0094 | 0.0384 | 0.0290 | 0.0715 | 0.0718 | 0.0179 |
| | q | r | s | t | u | v | w | x |
| $p$ | 0.0053 | 0.0561 | 0.0724 | 0.0820 | 0.0317 | 0.0169 | 0.0202 | 0.0049 |
| | y | z | | | | | | |
| $p$ | 0.0196 | 0.0023 | | | | | | |

The vector $\mathcal{G}(\mathcal{M})$ is

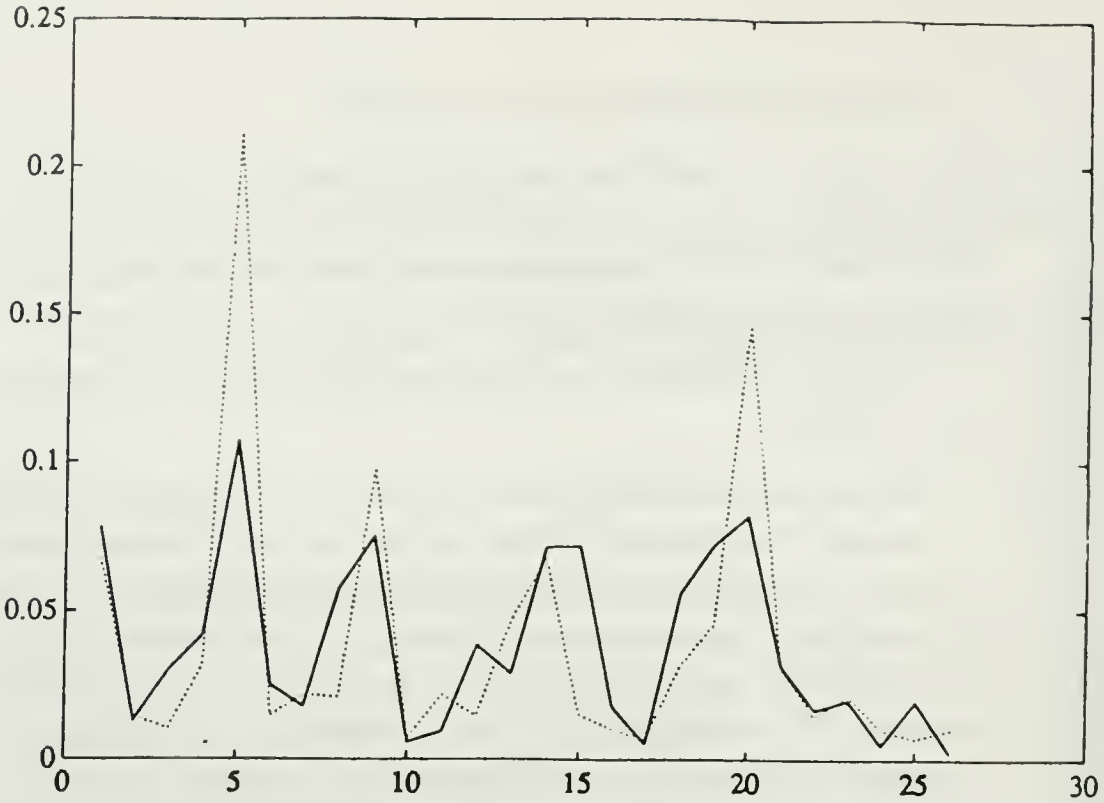



Figure 3.2: $\mathcal{T}(\mathcal{M})$ solid line; $\mathcal{G}(\mathcal{M})$: dotted line.

| | a | b | c | d | e | f | g | h |
|---|---|---|---|---|---|---|---|---|
| $p$ | 0.0679 | 0.0146 | 0.0101 | 0.0315 | 0.2112 | 0.0146 | 0.0218 | 0.0210 |
| | i | j | k | l | m | n | o | p |
| $p$ | 0.0979 | 0.0070 | 0.0218 | 0.0146 | 0.0471 | 0.0679 | 0.0152 | 0.0101 |
| | q | r | s | t | u | v | w | x |
| $p$ | 0.0070 | 0.0315 | 0.0454 | 0.1465 | 0.0315 | 0.0146 | 0.0218 | 0.0101 |
| | y | z | | | | | | |
| $p$ | 0.0070 | 0.0101 | | | | | | |

The cross entropy between the two is $K(\mathcal{T}(\mathcal{M}), \mathcal{G}(\mathcal{M})) = 0.1993$.

In Figure 3.2 we can compare the values of $\mathcal{T}(\mathcal{M})$ against the ones of $\mathcal{G}(\mathcal{M})$.

□

### 3.2.2 Information Mechanics

> "Computing processes are ultimately abstractions of physical processes: thus a comprehensive theory of computation must reflect in a stylized way aspects of the underlying physical world. On the other hand, physics itself may draw fresh insights and productive methodological tools from looking at the world as an ongoing computation. The term *information mechanics* seems appropriate for this unified approach to physics and computation."[82]

Let us try to consider the function that we want to minimize in a different light. If we multiply $K(\mathcal{T}(S), \mathcal{G}(S))$ by the constant $T = 1/\beta$ we can write it in the form

$$TK(\mathcal{T}(S), \mathcal{G}(S)) = T \sum q \log q + \sum q\,|s| = \langle |s| \rangle - TS$$

If we interpret cost as energy, we can see that there is a consistency between the two concepts at the *microscopic* and the *macroscopic* level and that our function takes in fact the familiar form of free energy (we have overloaded the symbol $S$ which on the right side of the equality stands for entropy density). This outcome is not surprising and it depends on the way we have constructed our reasoning. However in the present situation we know what "temperature" stands for, and we can attempt an interpretation at the microscopic level of the process of minimization as a process of thermalization.

If we consider $T$ as a function of the number of symbols $n$ we see that negative temperatures correspond to a number smaller than 1 of symbols, situation quite difficult to imagine in both directions (remember that T is the absolute temperature). The interesting case $T > 0$ shows that smaller temperatures correspond to a higher number of symbols. The process of cooling down the system in order to find the minimum of the cross-entropy function corresponds, at the microscopic level, to allowing the possibility to use more and more symbols and hence to distinguish more and more signals. We should notice that even if we let the temperature go to zero, we can reasonably expect the system to freeze on a configuration with a finite, relatively small, number of symbols: when the temperature is small it is unlikely to deviate too much from an already good configuration.

## 3.3 A Learning Rule for Neural Nets

The ideas we have considered so far can find an application to the study of neural nets. In this section we will propose an architecture which permits a representation of the subsequent encodings of a text as the transmission of a sensory signal through a hierarchical tissue. The learning Rule that we propose is derived by the findings in Section 3.2. A generalization in the flavor of Section 3.1 can have an analogous development.

### 3.3.1 The Logical Neural Tissue

The neural architecture described in this section is inspired by the optimizing principles derived earlier in this chapter and is grounded on the neurological fact that the reaction time to a sensory input is roughly equal to the time necessary for the signal to travel through the neural system. We propose the intuition that the elaboration of the input takes place during the transmission. As a first approximation we consider an architecture consisting of a hierarchy of layers of logical neurons. Each neuron transmits its excitation to neurons in the level above and receives excitation from the level below.

The units at level $L_0$ are the sensory units. We assume the net is *a priori* able to detect a set of features of the input. For each feature there exists a population of units which is triggered by its presence in the window.

Which one is the sensory level is obviously conventional. At each level, in fact, each unit is responding to higher level features formed by the concomitant presence of lower level ones. Hence at any level $L_i$ the units can be considered as functional receptors. It is just a matter of conventions, depending on what level of features in the sensory input the units are assumed to be able to respond to.

### 3.3.2 The Learning Rule

The transition of the signal from one layer to the succeeding one corresponds to one level of encoding. The learning rule that we will now describe operates unchanged

at all levels in the hierarchy. We describe it in general by explaining how to set the weights from the units at level $L_i$ to the units at level $L_{i+1}$.

In the following we assume that the threshold of every unit is 1. Let us consider every unit $s$ as a signal, and assign to it an "energy" $\varepsilon(s)$ equal to the information theoretical cost of transmitting the signal in an optimal $n-$ary code

$$\varepsilon(s) = -\frac{\log_e p(s)}{\beta} \tag{3.13}$$

where $\beta = \log_e n$ and $p(u)$ is the probability of firing for unit $u$ computed as normalized relative frequency with respect to the other units in the same level.

To every unit $t$ in level $L_{i+1}$ is assigned an energy equal to

$$\varepsilon(t) = \sum_{s \epsilon L_i} w_{s,t} \varepsilon(s) \tag{3.14}$$

where $w_{s,t}$ is the weight of the connection of $s$ to $t$. The intuitive justification of (3.14) is obvious: every unit is assigned an energy equal to the weighted sum of the energies of the units connected to it.

We enforce now the intuition that each layer can be considered as the sensory layer, so we want the neurons in layer $L_{i+1}$ to perform as symbols in an optimal code, so the learning process should set the weights $w_{s,t}$ in such a way that

$$p(t) = \frac{e^{-\beta \varepsilon(t)}}{Z} \tag{3.15}$$

$\forall t$, where $Z = \sum e^{-\beta \varepsilon(t)}$.

This rule is also justified by the following argument. The firing of each unit at level $L_i$ carries an energy that is transmitted by the units connected to it at level $L_{i+1}$. We want the unit at level $L_{i+1}$ behave, energetically, as a system in equilibrium with that at level $L_i$. This approach is founded on the assumption that any optimal semiotic system has to satisfy this requirement, as derived in the previous Section 3.2.

After the net has been trained on a particular homogeneous reality, it will show recognition of other instances of the same training set by being able to transmit the signals from the sensory units to higher levels of the hierarchy. Activations in high

level layers correspond to a high level description of the input. A sample from a radically different reality will result in little or no transmission of the signal.

### 3.3.3 Further Developments

One way to implement the learning rule of Section 3.3.2 to perform a simulated annealing algorithm [50] in order to minimize a suitable error function (for example $\left|p(t) - \frac{e^{-\beta\epsilon(t)}}{Z}\right|$). To speed up the process we can make the weight matrix sparse, that is each single unit is connected only to a small number $n$ of other units chosen at random.

There are many assignments to the weight matrix which satisfy condition (3.15). *A priori*, all of the assignments are equally adequate and they correspond to different *Weltanschauungen*. The possibility of comprising many different ones depends on the number of elements in the population detecting a particular feature and the number of connections. This explains the intuition that the higher the number of units and connections, the more versatile is the net.

In many fields of application, like natural language, the interesting feature of the reality under consideration is just the temporal correlation of the images present on the window at different instants of time. In such applications it is advisable to provide, at every level of the net, some units with a delayed reaction. Such units will provide a sort of short term memory.

Relying on the global situation for determining each single weight is the major limitation of our learning rule and we can easily expect the same slow convergence as in the Boltzman machine. Moreover our learning rule relies on the availability of the information about the frequency of firing of every unit. In some way yet another global mechanism, exterior to the net, has to provide that information. These limitations nullify any advantage of a neural net implementation over the algorithmic implementation of the process. Thus it is advisable to focus on the search for a local learning rule with the same desired global behavior and/or a distributed mechanism to keep the information regarding the relative frequency of every neuron updated.

The ideas exposed in 3.3.2 can also be exploited for the study of any general neural net. In fact, given any neural net with fixed weights, if we assign to each neuron of a net a symbol -that is we adopt the grandmother neuron viewpoint- the history of the net can be interpreted as a sequence of encodings of the sensory input. If we want those encodings to be optimal, then the net has to satisfy condition (3.15). It would be intersting to check how many of the learning rules present in the literature fix the weights so as to satisfy condition (3.15).

# Chapter 4

# Algorithms

Once a text has been inputed, and interpreted, we can change the level of perception and face the same problem again. It is immediately evident that the process of constructing an interpretational scheme to obtain a larger representation of $\mathcal{T}$ can be iterated. We can think of an hypothetical learning machine which constructs layers of interpretational schemes.

The object of the present chapter is to find a a global picture for this process, by describing some algorithms that concretize the abstract learning machine that we have outlined in Section 2.2.5.

## 4.1 Minimal Description

We have the alphabet as a universal model for structures, and its compression coefficient as a measure of its effectiveness on a given text. We are ready to describe our learning algorithms. As we have promised, our goal is to achieve an efficient, if not minimal, description of the text to be learned. So we need to introduce the following

**Definition 21** Let $B$ be an alphabet. The *size* of $B$ is $|B| = \sum_{s \in B} |\psi(s)|$

The *maximum size* of $B$ is $|B|_{\max} = \max_{s \in B} |\psi(s)|$

**Definition 22** Let $\mathcal{T}$ be a text over an alphabet $A$. A description of $\mathcal{T}$ is a pair $(B, \mathcal{T}')$ where $B$ is an alphabet over $A$ and $\mathcal{T}'$ is a text over $B$ such that $\overline{\mathcal{T}}^B = \mathcal{T}'$.

The *size of the description* $(B, \mathcal{T}')$ is

$$|(B, \mathcal{T}')| = |B| + |\mathcal{T}|\,\lambda_{\mathcal{T}}(B)$$

□

**Remark 6** Note that if the text is much bigger than the alphabet, then the only significant contribution in the size of the description is given by the second term $|\mathcal{T}|\,\lambda_{\mathcal{T}}(B)$.
□

## 4.2 Enumeration Algorithms

All the algorithms in this section take a text as input and return the topmost alphabet that they have constructed. They will represent different stages of evolution of the trivial enumeration algorithm, the universal program that solves everything.

We want to model learning, evolution in time, but still we want the algorithms to be algorithms, i.e. engines that consume input and produce output. So we make the following assumption: the algorithms take a text as input and produce an alphabet as output on-line right away. Then, they will start working and, little by little, update their output. The first step in this abstraction will be to assume that all of the text is inputed in one single operation. Later on we will reconsider this assumption and find a way to render the process more feasible.

### 4.2.1 Universal Enumeration

It is not difficult to see that there exists an effective enumeration of all the alphabets over a given alphabet, the reader can construct it easily by himself. The only things to be treated with a little bit of care are the tests for completeness and

**Algorithm 1**

Input: a text $\mathcal{T}$ on an alphabet $A_0$
Current output: an alphabet $A_1$ over $A_0$

Let $A_1 = A_0$
forall alphabets $B$ over $A_0$
    if $\left|(B, \overline{\mathcal{T}}^B)\right| < \left|(A_1, \overline{\mathcal{T}}^{A_1})\right|$ then $A_1 = B$.
end forall

Figure 4.1: Universal Enumeration

unambiguousness. Completeness of an alphabet is easily decided by brute force since it involves only a finite number of checks. However, the brute force test for unique decipherability would involve infinitely many trials. Luckily, there exist decidable necessary and sufficient conditions for unique decipherability [16].

Given all this, it is easy to devise an algorithm for learning that uses the size of the description as a heuristics. The algorithm is given in Figure 4.1

There could be very ill-natured infinite texts for which this algorithm never stabilizes. However, at any time, $A_1$ will be the alphabet among the ones the enumeration has constructed until then, which yields the minimal description $(A_1, \overline{\mathcal{T}}^B)$. The reader is urged to pause, consider this algorithm a little more closely, and try to answer the question: What is its running time?

The way things have been defined make this question particularly difficult, even in its formulation. Algorithm 4.2.1 gives an output right away, so the running time seems to be O(1). However that is not completely true: at the beginning the output is quite useless but it improves with time. So, how long does it take it to give *the best* output? Well, we know that there might not exist such a best[1]. Moreover, even

[1]It is a consequence of the non-computability of Kolmogorov complexity

if we restrict the inputs to those texts for which a best exists and the algorithm converges to the best in the limit, still the running time will depend on the input *structure*, which is not known until an output is produced. The discussion about running time looks interesting, but we will postpone it.

There is a more interesting observation to make about the amount of work done by this algorithm: whenever the output is changed, the algorithm has done a *maximum*[2] amount of work; it has tried everything possible before getting at the right spot.

If we now consider the behavior of Algorithm 4.2.1, we can't help noticing that it seems pretty dull. Whenever it finds a new alphabet that looks better than the previous one, it forgets about the work done before and gets so excited about the new result that it takes it as the best without giving it a second thought.

Despite these drawbacks, there is something very good about this algorithm: in the limit it yields an optimal alphabet and the efficiency of the output is monotonically nondecreasing.

### 4.2.2 Hierarchical Enumeration

Algorithm 4.2.1 presents all the problems that enumeration algorithms have: it is infeasible for real applications. As we have noticed, the amount of work that it does is maximum. So we ask the following question: Is it possible to save it some work?

Fortunately our model is "less abstract" then other models for structures and makes the algorithm amenable to improvements. We will be able to do this by providing to our algorithm, a little consciousness about the current best output, i.e. with a little bit of *memory* about the work done in the past, and by posing a bound on the maximum amount of work performed: instead of enumerating all of the alphabets, we will limit the enumeration to a small subset of them by putting a limit on their size. The algorithm is given in Figure 4.2

We should be precise about the algorithmic notation used. The first assumption is that the "loop $H$" construct is smart enough to find out if there has been a

[2]Maximum with respect to the enumeration chosen.

**Algorithm 2**

Input: an alphabet $A_0$, a text $\mathcal{T}$.
Current output: an alphabet $A_1$.

Let $A_1 = A_0$
loop H
    for all alphabets $B$ over $A_1$ such that $|B|_{\max} \leq h$
        if $\left|(B, \overline{\mathcal{T}}^B)\right| < \left|(A_1, \overline{\mathcal{T}}^{A_1})\right|$ then $A_1 = B$.
    end for all
end loop H

Figure 4.2: Hierachical Enumeration

progress, i.e. it incorporates a test to find out if $A_1$ has changed since the previous iteration. Second we should say something about the assignment $A_1 = B$. Note that $B$ is an alphabet over $A_1$ but we want $A_1$ to remain flat over $A_0$. So we assume that the code part of $B$ is a code over the code part of $A_1$ and that the classification part of $B$ is a classification over the classification part of $A_0$. Moreover whenever $s$ is formed as a string of the strings $s_1, ..., s_n$ then $\psi(s)$ is the concatenation $\psi(s_1)...\psi(s_n)$, analogously if a class c is formed as a class of $c_1, ..., c_m \in A_1$ then $c = \bigcup c_i$. in other words that assignment forgets the history of how the symbol has evolved.

We can consider $H$ and $h$ as inner parameter and call Algorithm 4.2.2, $U(h, H)$. It makes sense now to talk about the running time of $U$ since there are bounds, namely $H$ and $h$ which guarantee that it will eventually stop. Let us denote with $f$ its running time. It is obvious that $f$ depends on the text $\mathcal{T}$ (and on the number of symbols $n$ of the alphabet of $\mathcal{T}$), on the bound for the enumeration $h$, and on the number of levels $H$. Then $f(\mathcal{T}, h, H)$ and it is easy to see that $f(\mathcal{T}, h, 1) = O(2^{n^h})$ for every $\mathcal{T}$.

Let us observe that Algorithm 4.2.1 is the same as $U(\infty, 1)$. Hence it make sense to consider $h$ as the *accuracy* parameter, taken as efficiency of the representation found, and $H$ as the *speed* parameter in term of running time. In fact the number of texts for which the most efficient representation is found by $U(h, H)$ grows with $h$. On the other hand the maximum size of the alphabet constructed by $U(h, H)$ can get as large as $h^H$, in fact, the number of leaves of a tree with nodes with $h$ children and $H$ generations can grow as much as $h^H$. In order to understand better the idea of how it is possible to improve speed at the cost of optimality of the solution we should compare $U(h, 1)$ with $U(2, \log_2 h)$. The texts for which $U(h, 1)$ will find a structure while $U(2, \log_2 h)$ will not are all those that show all the possible digrams with the same frequencies but still with a structure at a higher level.

Let us say that we are not interested in such texts[3], but if we had to deal with them, it is possible to appropriately modify the situation to overcome the problem, for instance by dovetailing on $h$. So let us concentrate only on those text $\mathcal{T}$ such that $U(2, \log_2)$ and $U(h, 1)$ yield the same result, i.e, build the same alphabet.

In the light of what has been observed let us consider the algorithm in Figure , with the convention that the input to $U(h, H)$ are put in square brackets. note that the assignment statement now does not create any problem and that the same convention as before holds for the loop statement.

Let us call this algorithm $U_2$, again parametrized by $H$ and $L$. Note that the assumptions that we have made about the texts that we are considering, imply $U(h, 1) = U(2, \log_2 h) = U_2(\log_2 h, 1)$ so $U_2$ is still universal. $U_2$ exhibits a nice characteristic: when the search in the call for $U$ has stopped at the value $H$ and no more progress is possible, it allows to repeat the process starting on the new alphabet created. The interesting fact is that at any iteration of the outer loop the new alphabet will be atomic.

Consider the following example:

**Example 26** Let $S$ be the semigroup over the set of strings

$$\{aaaccc, bbbccc, aaabbb, cccccc\}$$

[3]Natural language texts don't have this property.

**Algorithm 3**

Input: an alphabet $A_0$, a text $\mathcal{T}$ on an alphabet $A_0$
Current output: an alphabet $A_L$
Variables: a hierarchical family of alphabets $\{A_i\}_{i=0}^{L}$

loop $L$

$A_L = A_{L-1}$

$A_L = U(2,H)[A_{L-1}, \overline{\mathcal{T}}^{A_{L-1}}]$

end loop $L$

Figure 4.3: Procrustes New Generation

and $\mathcal{T}$ any concatenations of words in $S$. Then $U(h,1)$ recognizes the structure of $\mathcal{T}$ *completely* only if $h \geq 6$, while $U(2,3)$ will do as well.

Note that the alphabet found by $U_2(1,3)$ is the same as the one found by $U(2,3)$, but $U_2(2,2)$ finds an alphabet which looks more natural.
□

Given all the discussion above, one can take values that the parameters of the algorithms take to learn a text as a characteristic of the text. We should emphasize that natural texts often show similar characteristics at different levels [11,65]. We have seen that there are ways to overcome to a certain extent the problems of enumerative learning algorithms provided the text exhibits nice properties. We can set forth the conjecture that natural languages are codes that evolved a structure that makes their learning "easy", i.e. the same convergence is attained with most search strategies. "Difficult" texts, like the ones structured on random sequences, are accidents that can happen only with the malicious intervention of a human being designing encodings *ad hoc*. It is reasonable to expect that they are not encountered when natural texts are considered.

**Algorithm 4**

Input: an alphabet $A_0$, a text $\mathcal{T}$ on an alphabet $A_0$
Current output: an alphabet $A_L$
Variables: a hierarchical family of alphabets $\{A_i\}_{i=0}^{L}$

loop $L$
    $A_c = A_L = A_{L-1}$
    mark all $A_c$ unvisited
    while $A_c$ not all visited
        choose an unvisited $s \in A_c$ for which $CONDITION$ holds
        let $D$ be the set of all 2-grams $st$ such that $t \in A_c$
        $A_{c'} = A_c \cup D - \{s\}$
        if $\lambda_{\overline{\mathcal{T}}^{A_L}}(A_c) > \lambda_{\overline{\mathcal{T}}^{A_L}}(A_{c'})$ then $A_c = A_{c'}$; mark all $A_c$ unvisited
        else mark $s$ visited
    end while
    $A_L = A_c$
end loop $L$

Figure 4.4: Heuristic Approach Algorithm

Apart from these considerations on natural languages, we could look more attentively into the abstract capabilities of Algorithm 4.2.2. It is true that the assumption that we made on the nature of the text that $U_2$ learns look strong at a first glance and are likely to leave out a lot of texts. Nonetheless we should remember the result [21] that of all the $n^h$ strings of length $h$ over an alphabet with $n$ symbols, only $O(h)$ are structured. Informally speaking, structured texts are relatively few.

### 4.2.3 Heuristic Approach

Despite the big improvements on Algorithm 4.2.1, Algorithm 4.2.2 still presents

a certain degree of enumeration. It is impossible to tell *a priori* what is the size of the alphabet that it will be enumerating on, since that is known only at the first iteration. So, in an actual applicative situation, it is probably a good idea to change the "forall" statement with a nondeterministic "choose" and then make the choice on the base of *a priori* heuristic judgements. Of course, if experimentation proves it worthwhile, nothing prevents the heuristic considerations on which they are based to be turned into proofs by means of a more attentive study.

The algorithm given in Figure 4.4 behaves essentially as Algorithm 4.2.2 but it avoids the enumeration by constructing an alphabet at the provisional level which *very likely* will provide a better description, that is a lower $\lambda$ coefficient.

*CONDITION* should be set appropriately. Reasonable conditions are, for instance

- $p(s)$ is minimum
- $p(x)p(s|x)$ is maximum

Anyway, the correct choice for *CONDITION* depends on the nature of the text and can be determined by experimentation.

## 4.3 Open Problems and Future Work

Thanks to their simplicity the algorithms in the previous section can be applied to many different areas. They can be the object of further study or they can be used as investigation tools. We will not enumerate all the possible way future research in the subject may proceed.

However there is another problem which is not immediately evident, on which we would like to draw attention.

### 4.3.1 A Characterization of Natural Languages

Let $\mathcal{T}$ be a text over $A_0$ and $\tilde{A}$ the code constructed by Algorithm 4.2.1 and $\{A_i^h\}_{i=0}^n$ the hierarchy of codes constructed by Algorithm 4.2.2, when the search is limited only to codes. Define $\mathcal{T}_i$ recursively as follows:

$$\mathcal{T}_0 = \mathcal{T}$$

$$\mathcal{T}_i = \overline{\mathcal{T}}_{i-1}^{A_i^h}$$

Let us make the additional assumption that $\mathcal{T}$ is such that $\overline{\mathcal{T}}_i$ is much bigger with respect to $\left|A_i^h\right|^h$, $|\mathcal{T}_i| \gg \left|A_i^h\right|^h$.

In this situation the only contribution to the size of the description $(A_i^h, \mathcal{T}_i)$ is given by $|\mathcal{T}_{i-1}|\, \lambda_{\mathcal{T}_{i-1}}(A_i^h)$

Consider again the hierarchy $\{A_i^h = (\overline{X_i}, \psi_i, A_{i-1}^h)\}_{i=1}^n$. Let $\psi_i(x) = s_1 ... s_k$ and define new codes $\{A_i' = (\overline{X_i}, \psi_i', A_0)\}_{i=1}^n$ recursively in the following way:

$$\psi_1'(x) = \psi_1(x)$$

$$\psi_i'(x) = \psi_{i-1}'(s_1) ... \psi_{i-1}'(s_k)$$

It is an interesting problem to characterize the texts $\mathcal{T}$ for which there exists a number $h$ such that

$$A_n' = \tilde{A}$$

For such texts the search for the optimal description would be linear in the number of levels, if $h$ is known. Notice that natural language seem to have a hierarchical organization where $h$ is small (about 5).

Another interesting problem is to see whether there exists an optimal value of $h$ which would would insure the right convergence of the Algorithm 4.2.2 for all structured texts[4], except possibly a "small" set of texts.

[4]See Introduction for an explanation of what "structured" means

# Bibliography


[1] D. Angluin and C.H. Smith, Inductive Inference: Theory and Methods, *Computing Surveys* 15 (1983), 237-268.

[2] J.F. Allen, C.R. Perrault, Analyzing Intention in Utterances. Artificial Intelligence 15 (1980), pp. 143-178.

[3] J.L. Austin, *How to do things with words*, Oxford Univ. Press, New York (1962).

[4] H.P.Barendregt, *The Lambda Calculus. Its Syntax and Semantics*, North-Holland, Amsterdam (1984).

[5] R. Berwick, *The Acquisition of Syntactic Knowledge*, MIT Press, Cambridge (1985).

[6] L. Blum and M. Blum, Towards a mathematical theory of Inductive Inference, *Inf. Contr.* 28 (1975), 125-155.

[7] E.R. Caianiello, Outline of a Theory of Thought Processes and Thinking Machines, *Jour. of Theoretical Biology* 1, 204 (1961).

[8] E.R. Caianiello, Some remarks on Organization and Structure, *Biol. Cybernetics 26* (1977), 151-158.

[9] E.R. Caianiello and R. Capocelli, On form and language: The Procustes Algorithm for feature extraction, *Kibernetik* 8 (1971), 223-233.

[10] E.R. Caianiello and R. Capocelli, Structural analysis of hierarchical systems, *Proc. 3rd Joint Conf. Pattern Recognition* (1976).

[11] E.R. Caianiello et al., Structure and Modularity in Self-Organizing Complex Systems, in *Topics in the General Theory of Structure* E.R. Caianiello and M.A. Aizenman eds., D. Reidel (1987).

[12] P. Caianiello, Inductive Inference and Encoding, Tech. Rep. 372, Comp. Scie. Dept., New York University (1988).

[13] P. Caianiello, Learning by Syllabation and Classification, Tech. Rep. 424, Comp. Scie. Dept., New York University (1988).

[14] P. Caianiello, Neural Models, Structure and Learning, *Proceedings of the First Italian Workshop on Parallel Architecture and Neural Nets 1988*, World Scientific (1989).

[15] P. Caianiello, Learning by Data Compression in Neural Nets, *Proceedings of the Second Italian Workshop on Parallel Architecture and Neural Nets 1989*, to appear, World Scientific.

[16] R.M. Capocelli, A Necessary and Sufficient Condition for unique decipherability, *Notices of A.M.S.*, 22 A 714,(1975).

[17] R.M. Capocelli and L.M. Ricciardi, A Heuristic Approach to Feature Extraction and Compression for Written Language, Tech. Rep. Serie III N. 125, Istituto per le Applicazioni del Calcolo, Roma (1978).

[18] R.M. Capocelli and I. Taneja, On Some Inequalities and Generalized Entropies: a Unified Approach, *Cybernetics and Systems 16*, 4 (1986) 341-376.

[19] G.H. Chaitin, A theory of program size complexity formally equivalent to information theory, *Jour. of the ACM* 22 (1971), 329-340.

[20] G.H Chaitin, *Algorithmic Information Theory*, Cambridge University Press, (1988).

[21] G.H. Chaitin, Randomness and Mathematical Proof, *Scientific American*, 232 (5) (May 1975).

[22] G.H. Chaitin, Towards a mathematical Definition of Life, in *The Maximum Entropy Formalism*, Levine and Tribus (eds), MIT Press, 477-498 (1979).

[23] Chaitin, G.J. Godel's Theorem and Information, *Int. Jour. of Theoretical Physics*, Vol 21 (12), 941-954, (1982).

[24] N. Chomsky, *Knowledge of Language*, Praeger, New York (1978).

[25] N. Chomsky, *Aspects of the Theory of Syntax*, MIT Press, Cambridge, (1965).

[26] P.M. Cohn, *Universal Algebra*, Reidel (1981).

[27] Cohen, P.R., Perrault, C.R., Allen J.F. Beyond Question Answering. In *Strategies for Natural Language Processing*, W lenhert and M. Ringle, Eds. Lawerence Erlbaum Assoc., Hillsdale, NJ, (1982)

[28] C.M.Cook, A. Rosenfield, and A.R. Aronson, Grammatical inference by hill climing, *Information Sciences* 10 (1976), 59-80.

[29] T. Cover, Kolmogorov complexity, data compressing, and inference, in *The Impact of Processing Techniques on Communications*, J.K.Skwirzynski ed., Martinus Nijhoff Publisher (1985).

[30] M.D. Davis and E.J. Weyuker, *Computability, Complexities, and Languages* Academic Press, New York (1983).

[31] A. De Luca, On the Entropy of Formal Languages, in *Lecture Notes in Computer Science* 33 (1975), 103-109.

[32] T. De Mauro *Minisemantica*, Laterza, Roma (1982).

[33] A. De Santis, G. Markowsky and M.N. Wegman, Learning Probabilistic Prediction Functions, *Proc. of the 29th Symposium of FOCS* (1988), 110-119.

[34] De Saussure, *F. Course de linguistic generale*, Italian translation, Laterza, Rome, (1962).

[35] R.O.Duda and P.E. Hart, *Pattern Classification and Scene Analysis*, Wiley (1973).

[36] G. Dunn, B.S. Everitt, *An Introduction to Mathematical Taxonomy*, Cambridge Univ. Press (1982).

[37] I.J. Gelb, *A Study of Writing*, Univ. of Chicago Press (1963).

[38] E.M. Gold, Language Identification in the Limit, *Information and Control*, 10, 447-474.

[39] R. Grishman, *Computational Linguistics: an Introduction*, Cambridge University Press (1986).

[40] H. Haken, *Information and Seforganization*, Springer-Verlag (9188).

[41] P.R. Halmos, How to Write Mathematics, *L'Enseignement matematique*, T. XVI, fasc. 2 (1970).

[42] Z. Harris, String Analysis and Sentence Structure, Mouton & Co., The Hague (1962).

[43] Z. Harris, *Mathematical Structure of Language*, Wyley Inetrscience, New York (1968).

[44] G.W.Hart, *Minimum Information Estimation of Structure*, Ph.D. Thesis, M.I.T. (1987).

[45] G. Herdan, *Language as Choice and Chance*, Nordhoff (1956).

[46] J.E. Hopcroft and J.D. Ullman, *Introduction to Automata Theory, Languages and Computation*, Addison-Wesley (1979).

[47] N. Jacobson, *Basic Algebra I*, Freeman (1974).

[48] E.T. Jaynes, Information Theory and Statistical Mechanics I, *Phys. Rev. 106* (1957), 620-630.

[49] A.I. Khinchin, *Mathematical Foundations of Informiaton Theory*, Dover, New York (1957).

[50] S. Kirkpatrick, C.D. Gelatt Jr., M.P. Vecchi, Optimizing by Simulated Annealing, *Science 220* (1983), 671-680.

[51] A.N. Kolmogorov, Three Approaches to the quantitative Definition of Information, *Prob. Inf. Trans.* 1 (1965), 1-7.

[52] M. Koppel, Structure, *manuscript* (1987).

[53] S. Kullback, *Information Theory and Statistics*, Wiley (1959).

[54] L.D. Landau, E.M. Lifshitz, *Statistical Physics*, Addison-Wesley (1969).

[55] S.K. Leung-Yang-Cheong and T. Cover, Some equivalences between Shannon entropy and Kolmogorov complexity, *IEEE Trans. on Information Theory* 24 (1978), 331-337.

[56] G.C. Lepschy *La linguistica strutturale*, Einaudi , Torino (1966).

[57] A. Levy, *Basic Set Theory*, Springer-Verlag (1979).

[58] D.G. Lockwood, *Introduction to Stratification Linguistics*, Harcourt Brace Jovanovich (1972).

[59] A.R. Luria, *The Working Brain, An Introduction to Neuropsychology*, Basic Books (1973).

[60] Mandelbrot, B. Structure Formelle Des Textes et Communication, *Word* 10, 1-27, (1954).

[61] B. Mandelbrot, On the Theory of Word Frequencies and on Related Markovian Models of Discourse, in *Structure of Language and its Mathematical Aspects*, R. Jakobson ed., Am. Math. Soc. (1961).

[62] M. Mezard, G. Parisi, M.A. Virasoro, *Spin Glass Theory and Beyond*, World Scientific (1987).

[63] G. Mounin *Clefs pour la semantique* ital. transl. Feltrinelli (1975).

[64] T.E. Moore *Cognitive Development and the Acquisition of Language*, Academic Press (1973)

[65] A. Negro et al., Hierarchy and Modularity in Natural Languages, *Topics in the General Theory of Structure* E.R. Caianiello and M.A. Aizerman eds., D. Reidel (1987)

[66] D.N. Osherson *et al.*, *Systems that Learn: an Introduction to Learning Theory for Cognitice and Computer Scientists*, MIT Press (1986).

[67] M. Piattelli-Palamarini, *Language and Learning: the debate between J. Piaget and N. Chomsky*, Harvard University Press (1980).

[68] Pinker S. Formal models of Language Learning, *Cognition* 7 , 271-283, (1979).

[69] J. Rissanen, Modeling by shortest data description, *Automatica* 14 (1978) 465-471.

[70] J. Rissanen and G.G. Langdon Jr, Universal modeling and coding,*IEEE Trans. on Info. Th.* IT-27 (1981) 12-23.

[71] J. Rissanen, A universal data compression system, *IEEE Trans. on Info. Th.* IT-29 (1983), 656-664 .

[72] J. Rissanen, Universal Coding, Information, Prediction and Estimation, *IEEE Trans. on Info. Th.* IT-30 (1984), 629-636.

[73] J. Rothstein, Information, Measurement and Quantum Mechanics, *Science 114* (1951).

[74] N. Sager, *Natural Language Information Processing*, Addison-Wesley, Reading, Mass. (1981).

[75] J.T. Schwartz *et al.*, *Programming with Sets: an Introduction to SETL*, Springer (1986).

[76] J.R. Searle, *Speech Acts.* Cambridge Univ. Press, New York, 1969.

[77] M.A. Selfridge, Computer Model of Child Language Learning, *Artificial Intelligence 29,* 171-216, (1986).

[78] C.E. Shannon, W. Weaver, *The Mathematical Theory of Communication*, University of Illinois Press, Urbana (1949).

[79] J.E.Shore, R.W. Johnson, Axiomatic Derivation of the Principle of Maximum Entropy and the Principle of Minimum Cross-Entropy, *IEEE Trans. on Information Theory 26* (1980), 26-37.

[80] J.E. Shore, R.W. Johnson, Properties of Cross-Entropy Minimization, *IEEE Trans. on Information Theory 27* (1981), 472-482.

[81] R.J. Solomonoff, A formal theory of Inductive Inference Part 1 and 2, *Info. and Control* 7 (1964), 1-22, 224-254.

[82] T. Toffoli, Physics and Computation, *International Journal of Theoretical Physics 21* (1982), 165-175.

[83] L.G. Valiant A Theory of the Learnable, *Comm. of the A.C.M. 27* 11 (1984) .

[84] N. Wiener, *Ex Prodigy: My Childhood and Youth*, Simon and Schuster (1953)

[85] J. Ziv and A. Lempel, On the Complexity of a Finite Sequences, *IEEE Trans. of Info. Th.* IT.21 1 (1976).

[86] J. Ziv and A. Lempel, A universal algorithm for sequential data compression, *IEEE Trans. on Info Th.* IT-23 (1976), 75-81.

[87] A.K. Zvonkin and L.A. Levin,The complexity of finite objects and the development of the concepts of information and randomness by means of the theory of algorithms, *Russian Mathematical Surv.* 25 (1970), 83-124.